Express Mail Label No. EV676909336US
Attorney Docket No. 283-205.03 Con
PATENT

SUBSTITUTE SPECIFICATION
REPROGRAMMABLE OPTICAL READER

Cross References to Related Applications

[0001] This is a divisional of U.S. Patent Application No. 09/385,597 filed on August 30, 1999, which is a continuation-in-part of U.S. Patent Application No. 08/839,020 filed April 23, 1997, which issued as U. S. Patent No. 5,965,863, which, in turn, is a continuation-in-part of U.S. Patent Application No. 08/697,913, filed September 3, 1996, which issued as U.S. Patent No. 5,900,613 on May 4, 1999, the contents of which are relied upon and incorporated herein by reference in its entirety, and the benefit of priority under 35 U.S.C. §120 is hereby claimed.

Background of the Invention

Field of the Invention

[0002] The present invention relates to hand held optical reading devices, and is directed more particularly to a hand held optical reading device configured to be programmed with use of a host processor.

Description of the Prior Art

[0003] One-dimensional optical bar code readers are well known in the art. Examples of such readers include readers of the SCANTEAM® 3000 Series manufactured by Welch Allyn, Inc. Such readers include processing circuits that are able to read one-dimensional (1D) linear bar code symbologies, such as the UPC/EAN code, Code 39, etc., that are widely used in supermarkets. Such 1D linear symbologies are characterized by data that is encoded along a single axis, in the widths of bars and spaces, so that such symbols can be read from a single scan along that axis, provided that the symbol is imaged with a sufficiently high resolution along that axis.

[0004] In order to allow the encoding of larger amounts of data in a single bar code symbol, a number of 1D stacked bar code symbologies have been developed, including Code 49, as described in U.S. Patent No. 4,794,239 (Allais), and PDF417, as described in U.S. Patent No. 5,340,786 (Pavlidis, et al). Stacked symbols partition the encoded data into multiple rows, each including a respective 1D bar code pattern, all or most all of which must be scanned and decoded, then linked together to form a complete message. Scanning still requires relatively high resolution in one dimension only, but multiple linear scans are needed to read the whole symbol.

[0005] A third class of bar code symbologies, known as two dimensional (2D) matrix symbologies, have been developed which offer orientation-free scanning and greater data densities and capacities than their 1D counterparts. Two-dimensional matrix codes encode data as dark or light data elements within a regular polygonal matrix, accompanied by graphical finder, orientation and reference structures. When scanning 2D matrix codes, the horizontal and vertical relationships of the data elements are recorded with about equal resolution.

[0006] In order to avoid having to use different types of optical readers to read these different types of bar code symbols, it is desirable to have an optical reader that is able to read symbols of any of these types, including their various subtypes, interchangeably and automatically. More particularly, it is desirable to have an optical reader that is able to read all three of the above-mentioned types of bar code symbols, without human intervention, *i.e.*, automatically. This in turn, requires that the reader have the ability to automatically discriminate between and decode bar code symbols, based only on information read from the symbol itself. Readers that have this ability are referred to as "autodiscriminating" or having an "autodiscrimination" capability.

[0007] If an autodiscriminating reader is able to read only 1D bar code symbols (including their various subtypes), it may be said to have a 1D autodiscrimination capability. Similarly, if it is able to read only 2D bar code symbols, it may be said to have a 2D autodiscrimination capability. If it is able to read both 1D and 2D bar code symbols interchangeably, it may be said to have a 1D/2D autodiscrimination capability. Often, however, a reader is said to have a 1D/2D autodiscrimination capability even if it is unable to discriminate between and decode 1D stacked bar code symbols.

[0008] Optical readers that are capable of 1D autodiscrimination are well known in the art. An early example of such a reader is the Welch Allyn SCANTEAM® 3000, manufactured by Welch Allyn, Inc.

[0009] Optical readers, particularly hand held optical readers, that are capable of 1D/2D autodiscrimination are less well known in the art, since 2D matrix symbologies are relatively recent developments. One example of a hand held reader of this type which is based on the use of an asynchronously moving 1D image sensor, is described in copending, commonly assigned U. S. Patent Application No. 08/504,643, now U.S. Patent No. 5,773,806, which application is hereby expressly incorporated herein by reference. Another example of a hand held reader of this type which is based on the use of a stationary 2D image sensor, is described in copending, commonly assigned U.S. Patent Application No. 08/914,883, now U. S. Patent No. 5,942,741, which is also hereby expressly incorporated herein by reference.

[00010] Optical readers, whether of the stationary or movable type, usually operate at a fixed scanning rate. This means that the readers are designed to complete some fixed number of scans during a given amount of time. This scanning rate generally has a value that is between 30 and 200 scans/sec for 1D readers. In such readers, the results of successive scans are decoded in the order of their occurrence.

[00011] Prior art optical readers operate relatively satisfactorily under conditions in which the data throughput rate, or rate at which data is scanned and decoded, is relatively low. If, for example, the scanning rate is relatively low and/or the data content of the bar code or other symbol is relatively small, *i.e.*, the scanner is operating under a relatively light decoding load, the decoding phase of the reading process can be completed between successive scans. Under these conditions scan data can be accurately decoded without difficulty.

[00012] Readers of the above-described type have the disadvantage that, if they are operated under relatively heavy decoding loads, *i.e.*, are required to rapidly scan symbols that have a relatively high data content, the tracking relationship or synchronism between the scanning and decoding phases of the reading process will break down. This is because under heavy decoding loads the decoding phase of a read operation takes longer than the scanning phase thereof, causing the decoding operation to lag behind the scanning operation. While this time lag can be dealt with for brief periods by storing the results of successive scans in a scan memory and decoding the results of those scans in the order of their occurrence when the decoder becomes available, it cannot be dealt with in this way for long. This is because, however large the scan memory, it will eventually overflow and result in a loss of scan data.

[00013] One set of solutions to the problem of maintaining the desired tracking relationship between the scanning and decoding phases of the reading process is described in previously mentioned copending U.S. Patent Application No. 08/914,883, now U. S. Patent No. 5,942,741. Another set of solutions to the problem of maintaining the desired tracking relationship between the scanning and decoding phases of the reading process is described in U.S. Patent No. 5,463,214, which issued on the parent application of the last mentioned copending patent application.

[00014] Generally speaking, the latter of these two sets of solutions to the above-discussed tracking problem involves the suspension of scanning for brief periods in order to assure that the scanning process does not pull too far ahead of the decoding process. The former of these two sets of solutions to the above-discussed tracking problem, on the other hand, involves the skipping over of one or more sets of scan data, in favor of more current scan data, if and to the extent necessary for tracking purposes, in combination with the use of two or more scan data memories to minimize the quantity of scan data that is skipped.

[00015] In the past, no consideration has been given to accomplishing scan-decode tracking in conjunction with 1D/2D autodiscrimination, *i.e.,* as cooperating parts of a single coordinated process. This is in spite of the fact that the 1D/2D autodiscrimination is known to involve heavy decoding loads of the type that give rise to tracking problems. Thus, a need has existed for an optical reader that combines a powerful tracking capability with a powerful 1D/2D autodiscrimination capability.

[00016] As new and/or improved 1D and 2D bar code symbologies, and as additional 1D and 2D decoding programs come into widespread use, previously built optical readers may or may not be able to operate therewith. To the extent that they cannot operate therewith, such previously built optical readers will become increasingly obsolete and unusable.

[00017] In the past, the problem of updating optical readers to accommodate new bar code symbologies and/or new decoding programs has been dealt with by manually reprogramming the same. One approach to accomplishing this reprogramming is to reprogram a reader locally, *i.e.,* on-site, by, for example, replacing a ROM chip. Another approach to accomplishing this reprogramming is to return it to the manufacturer or his service representative for off-site reprogramming. Because of the expense of the former and the time delays of the latter, neither of these approaches may be practical or economical.

[00018] The above-described problem is compounded by the fact that, if an optical reader is not equipped to operate as a tracking reader, it may not be possible to reprogram it to use an autodiscrimination program that is designed to be executed in conjunction with tracking. This is because the autodiscrimination program may include steps that require the tracking feature to prevent data from overflowing the scan memory and being lost. Alternatively, the scan rate may be decreased, although this reduction will adversely affect performance when low data content symbols are read. Thus, a need has existed for an optical reader that can be reprogrammed economically in a way that allows it to realize the full benefit of the 1D/2D autodiscrimination and tracking features, among others.

Summary of Invention

[00019] A programmable optical reader in one embodiment can include a program loading component and a program execution component operative to execute an externally generated program, whereby executing the externally generated program includes replacing a portion of the optical reader program. A programmable optical reader in another embodiment can include a two-dimensional image sensor, an image frame memory storing two-dimensional electronic images, and can be configured to be reprogrammed by any one of receipt of reprogramming data from a local host processor or receipt of programming data from an external remote off-site processor.

Description of the Drawings

[00020] Other objects and advantages will be apparent from the following description and drawings, in which:

[00021] Fig. 1 is a block diagram of an embodiment of the reading apparatus which is generic to reading apparatuses which utilize 1D and 2D image sensors;

[00022] Figs. 2 and 3 are block diagrams of embodiments of the reading apparatus which utilize 2D and 1D image sensors, respectively;

[00023] Figs. 4A, 4B, and 4C are oblique or partially cutaway views of the 2D reading apparatus of Fig. 2;

[00024] Figs. 4D, 4E, and 4F are oblique or partially cutaway views of an alternative embodiment of the reader apparatus of Fig. 2;

[00025] Figs. 4G, 4H, and 4I are oblique or partially cutaway views of another alternative embodiment of the reader apparatus of Fig. 2;

[00026] Figs. 5A, 5B, and 5C are oblique or partially cutaway views of the 1D reading apparatus of Fig. 3;

[00027] Fig. 6A is a flow chart of the main program of the reading apparatus;

[00028] Fig. 6B is a flow chart of a modified main program of the reading apparatus;

[00029] Fig. 7A shows the structure of one embodiment of a menu word or message suitable for use with the program of Fig. 6;

[00030] Figs. 7B and 7C are tables showing examples of the usages to which various parts of the menu word of Fig. 7A may be put;

[00031] Fig. 8 is a flow chart of the menu routine shown in Fig. 6;

[00032] Figs. 8A - 8D are examples of option symbol selection charts which may be used with the menuing feature;

[00033] Fig. 9 is a block diagram of a typical system with which the reading apparatus may be used;

[00034] Fig. 10A is a flow chart of a loading routine suitable for use;

[00035] Fig. 10B is a flow chart of a reprogramming routine suitable for use;

[00036] Fig. 11A is a flow diagram illustrating a primary program for a host processor configured for reprogramming of, and for other interactions with an optical reader;

[00037] Fig. 11B is a flow diagram illustrating a subprogram for reprogramming an optical reader in communication with a host processor;

[00038] Fig. 11C is a memory map for a memory space having stored thereon an operating program comprising a main program and a parameter table;

[00039] Fig. 11D is a flow diagram for a subprogram executed by a host processor for editing a parameter table;

[00040] Fig. 11E illustrates an exemplary parameter configuration screen;

[00041] Fig. 11F illustrates a flow diagram executed by a host processor for simulating the results of applying editing commands to a decoded message.

[00042] Fig. 12 is a timing diagram which shows the scanning/decoding relationship used by the prior art;

[00043] Fig. 13A through 13E are timing diagrams which illustrate various ones of the tracking relationships made possible;

[00044] Fig. 14 shows examples of memory structures that may be used in implementing the tracking relationships shown in Figs. 13A through 13E;

[00045] Fig. 15 is a simplified flow chart which illustrates the "Repeat Until Done", "Repeat Until Stopped", and "One Shot" scanning-decoding modes;

[00046] Fig. 16 is a flow chart of one embodiment of the 1D portion of the autodiscrimination program;

[00047] Figs. 17A through 17E are drawings which facilitate an understanding of the flow chart of Fig. 16;

[00048] Fig. 18 is a flow chart of one embodiment of the 2D portion of the autodiscrimination process;

[00049] Figs. 19A through 19D show representative bar code symbols of types that may be decoded by the reading apparatus; and

[00050] Fig. 20 is a flow chart that illustrates the effect of the code options of the autodiscrimination process.

Description of the Preferred Embodiments

[00051] Referring to Fig. 1 there is shown a block diagram of an optical reader 10. As will be explained more fully later, Fig. 1 shows the basic structures that together comprise the

general form of an optical reader that is suitable for use, and is generic to optical readers that use 1D image sensors and to optical readers that use 2D image sensors. Similarly, Fig. 2 shows the basic structures that together comprise the general form of optical readers that use 2D image sensors. Finally, Fig. 3 shows the basic structures that together comprise the general form of optical readers that use 1D image sensors. It will be understood that, except where specifically limited to readers having 2D or 1D image sensors, the present description refers generically to readers of any of the types shown in Figs. 1, 2, and 3.

[00052] Referring first to Fig. 1, the optical reader includes an illumination assembly 20 for illuminating a target object T, such as a 1D or 2D bar code symbol, and an imaging assembly 30 for receiving an image of object T and generating an electrical output signal indicative of the data optically encoded therein. Illumination assembly 20 may, for example, include an illumination source assembly 22, such as one or more LEDs, together with an illuminating optics assembly 24, such as one or more reflectors, for directing light from light source 22 in the direction of target object T. Illumination assembly 20 may be eliminated, if ambient light levels are certain to be high enough to allow high quality images of object T to be taken. Imaging assembly 30 may include an image sensor 32, such as a 1D or 2D CCD, CMOS, NMOS, PMOS, CID or CMD solid state image sensor, together with an imaging optics assembly 34 for receiving and focusing an image of object T onto image sensor 32. The array-based imaging assembly shown in Fig. 2 may be replaced by a laser array or laser scanning based imaging assembly comprising a laser source, a scanning mechanism, emit and receive optics, a photodetector and accompanying signal processing circuitry.

[00053] Optical reader 10 of Fig. 1 also includes programmable control means 40 which preferably comprises an integrated circuit microprocessor 42 and an application specific integrated circuit or ASIC 44. Processor 42 and ASIC 44 are both programmable control devices which are able to receive, output and process data in accordance with a stored program stored in either or both of a read/write random access memory or RAM 45 and an erasable read only memory or EROM 46. Processor 42 and ASIC 44 are also both connected to a common bus 48 through which program data and working data, including address data, may be received and transmitted in either direction to any circuitry that is also connected thereto. Processor 42 and ASIC 44 differ from one another, however, in how they are made and how they are used.

[00054] More particularly, processor 42 is preferably a general purpose, off-the-shelf VLSI integrated circuit microprocessor which has overall control of the circuitry of Fig.1, but which devotes most of its time to decoding image data stored in RAM 45 in accordance with program data stored in EROM 46. Processor 44, on the other hand, is preferably a special purpose VLSI integrated circuit, such as a programmable logic or gate array, which is programmed to devote its time to functions other than decoding image data, and thereby relieve processor 42 from the burden of performing these functions.

[00055] The actual division of labor between processors 42 and 44 will naturally depend on the type of off-the-shelf microprocessors that are available, the type of image sensor which is used, the rate at which image data is output by imaging assembly 30, etc. There is nothing in principle, however, that requires that any particular division of labor be made between processors 42 and 44, or even that such a division be made at all. This is because special purpose processor 44 may be eliminated entirely if general purpose processor 42 is fast enough and powerful enough to perform all of the functions contemplated. It will, therefore, be understood that neither the number of processors used, nor the division of labor therebetween, is of any fundamental significance.

[00056] With processor architectures of the type shown in Fig. 1, a typical division of labor between processors 42 and 44 will be as follows. Processor 42 is preferably devoted primarily to the tasks of decoding image data, once such data has been stored in RAM 45, handling the menuing options and reprogramming functions, and providing overall system level coordination. Processor 44 is preferably devoted primarily to controlling the image acquisition process, the A/D conversion process and the storage of image data, including the ability to access memories 45 and 46 via a DMA channel. Processor 44 may also perform many timing and communication operations. Processor 44 may, for example, control the illumination of LEDs 22, the timing of image sensor 32 and an analog-to-digital (A/D) converter 36, the transmission and reception of data to and from a processor external to reader 10, through an RS-232 (or other) compatible I/O device 37 and the outputting of user perceptible data via an output device 38, such as a beeper, a good read LED and/or a display 39 which may be, for example, a liquid crystal display. Control of output, display and I/O functions may also be shared between processors 42 and 44, as suggested by bus driver I/O and output/display devices 37′ and 38′ or may be duplicated, as suggested by microprocessor serial I/O ports 42A and 42B and I/O and display devices 37" and 38'. As explained earlier, the specifics of this division of labor is of no significance.

[00057] Referring to Fig. 2, there is shown a block diagram of an optical reader which is similar to that of Fig. 1, except that it includes optical and/or electrical assemblies and circuits that are specifically designed for use with a 2D image sensor. Accordingly, the optical and electrical assemblies and components of Fig. 2 are labeled with the same numbers used in Fig. 1, except for the addition of the suffix "-2". For example, image sensor 32-2 of Fig. 2 is a 2D image sensor which corresponds to generic image sensor 32 of Fig. 1, imaging optics assembly 34-2 of Fig. 2 is a 2D imaging optics assembly which corresponds to generic imaging optics assembly 34 of Fig. 1, and so on. In other words, corresponding elements of Figs. 1 and 2 have corresponding functions, although they may have different shapes and part numbers. Provided that these differences are taken into account, however, the description of the reader of Fig. 1 is equally applicable to the reader of Fig. 2, and will not be repeated herein.

[00058] One specific practical example of an optical reader of the type shown in Fig. 2 may be constructed using the particular commercially available solid-state integrated circuits listed in the following component table:

COMPONENT TABLE - Fig. 2	
Block Diagram Item	**Manufacturer/Part Number**
Image Sensor 32-2	VVL 1060B+
Prog. Gate Array 44-2	Actel 814V40A
Microprocessor 42-2	IDT 3081
EROM 46-2	Intel 28F400VB-B60
RAM 45-2	Toshiba TC51V4265DFT-60

[00059] Referring to Fig. 3, there is shown a block diagram of an optical reader which is also similar to that of Fig. 1, except that it includes optical and/or electrical assemblies and circuits that are specifically designed for use with a 1D image sensor. Accordingly, the optical and electrical assemblies and components of Fig. 3 are labeled with the same numbers used in Fig. 1, except for the addition of the suffix "-3". For example, image sensor 32-3 of

Fig. 3 is a 1D image sensor which corresponds to generic image sensor 32 of Fig. 1, imaging Optics assembly 34-3 of Fig. 3 is a 1D imaging optics assembly which corresponds to generic imaging optics assembly 34 of Fig. 1, and so on. Provided that these differences are taken into account, however, the description of the reader of Fig. 1 is equally applicable to the reader of Fig. 3, and will not be repeated herein.

[00060] One specific practical example of an optical reader of the type shown in Fig. 3 may be constructed using the particular solid-state circuits listed in the following component table:

COMPONENT TABLE - Fig. 3	
Block Diagram Item	**Manufacturer/Part Number**
Image Sensor 32-3	Toshiba 1201
Prog. Gate Array 44-3	Welch Allyn 21203276-01
Microprocessor 42-3	Motorola HC11
EROM 46-3	Atmel AT 29C257
RAM 45-3	Sony CXK 5864-BM-10LL

[00061] Significantly, the above-mentioned structural correspondences between Figs. 1, 2, and 3 should not be confused with the types of symbols that may be read thereby. More particularly, the 2D embodiment of Fig. 2 may be used to scan and decode both 1D and 2D bar code symbols. This is because both types of symbols can be imaged by a 2D image sensor. Similarly, the 1D embodiment of Fig. 3 may also be used to scan and decode both 1D and 2D bar code symbols. This is because a 1D image sensor may be used to image a 2D bar code symbol, provided that it is physically moved thereacross during the course of a scan. Because imaging of the latter type is described in detail in copending U.S. Patent Application No. 08/504,643, now U. S. Patent No. 5,773,806, which has been incorporated by reference herein, that type of imaging assembly will not be discussed again in full herein.

[00062] The reader structures shown in Fig. 2 are preferably supported on one or more printed circuit boards (not shown) that are, in turn, supported within a housing.

[00063] Examples of types of housings which may be employed to house elements of the reader apparatus shown in Fig. 2 are shown in Figs. 4A-4I. Figs. 4A-4C show a first exemplary housing 50-2-1, Figs 4D-4F show a second exemplary housing 50-2-2, while Figs.

4G-4I show a third exemplary housing 50-2-3. Housings 50-2-1, 50-2-2, and 50-2-3 are preferably shaped so as to fit comfortably into a human hand, and to include a finger actuatable trigger, 52-2-1, 52-2-2, 52-2-3. Housing 50-2-3 is shown as having an auxiliary trigger 52-2-3' which may supplement or replace trigger 52-2-3. Housings 50-2-1 and 50-2-2 have extending therefrom multiconductor cable or tether 54-2-1, 54-2-2, for providing communication with a local host processor, whereas 50-2-3 housing has extending therefrom an antenna 55-2-3 for providing a communication with a local host processor. It is seen further that housings 50-2-2 and 50-2-3 have incorporated therein displays 56-2-2, 56-2-3, for displaying information to a user, and a keyboard 58-2-2, 58-2-3, for inputting data and commands to processor 40.

[00064] Figs 5A - 5C show a housing 50-3 suitable for housing a 1D reader apparatus of the type described with reference to Fig. 3. Housing 50-3 includes a finger-actuatable trigger 52-3 and has extending therefrom a cable 54-3 for providing communication with a local host processor. Although not shown as containing such features, it is understood that housing 50-3 could readily be modified to include a display and a keyboard similar to those of 2D reader housings 50-2-2 and 50-2-3.

Main Program

[00065] The overall operation of the reader of Fig. 1 will now be described with reference to the flow chart of Fig. 6A. As will be explained more fully presently, Fig. 6A comprises a high level flow chart which illustrates the preferred embodiment of the main program of a reader which uses the apparatus and method. By "main program" is meant the program that illustrates the relationships between the major subdivisions or subroutines that together implement the above-described features. It also means the program that illustrates the overall flow and sequence of operations that are responsible for the advantages produced. Because Fig. 6A depicts the operation of two processors 42 and 44, however, operations that appear to be occurring sequentially may actually be occurring "simultaneously". Processor 44 may, for example, be imaging and storing newly scanned blocks of image data in RAM 45 while processor 42 is decoding blocks of image data that were stored in RAM 45 during earlier scans. This is possible because the two processors are operating in different memory spaces, in different time slots, or under the common control of a bus arbitration device. As a result, while the processors can never use the same memory or address space at the same time for conflicting purposes, they can be made to execute their respective programs sufficiently cooperatively and contemporaneously that they are effectively operating simultaneously. It is in this sense that the word "simultaneous" will be used herein.

[00066] Referring to Fig. 6A, the main program begins with block 605 which causes the reader to wait in a low power state until trigger 52 is pulled. When the trigger is pulled, the processor is directed to block 610, which causes it to power up and initialize the reader hardware, including the ASIC, the DMA channel and the I/O devices, among others. The processor is then directed to blocks 615 and 620 which cause it to define the image data memory space that will be used (block 615) and to initialize the reader with the default values of the operating parameters stored in the parameter table thereof (block 620).

[00067] The parameter table, which is preferably stored in EROM 46, specifies the values of the parameters that define the mode in which the reader will operate. Examples of these parameters include the size and the frame rate of the image sensor, the codes that will be enabled during autodiscrimination, the I/O communication protocols, beeper pitch or volume, among others. The default values of these parameters are those which will be used if the user or an externally generated reprogramming command does not specify other values, and correspond to a combination of parameters which are suitable for use under most operating conditions. The different parameters that may be used, and the affect that they have on the operation of the reader will be discussed in detail later.

[00068] After the reader has been initialized, the processor proceeds to blocks 625 and 627, which call for it to capture and attempt to decode an image of the target symbol. This involves the performance of a number of related steps, the particulars of which are determined by the parameters of the parameter table. Included among these steps are a scanning subroutine which specifies the address space or spaces in which scan data will be stored and whether scanning is to be continuous (*e.g.*, at a full video rate, such as 30 frames per second), or discontinuous (*e.g.*, with pauses related to the current state of the trigger). The operation of the decoding routine, which is executed in a user or factory selectable relationship to the scanning routine, is governed by parameters which control the codes which are enabled for processing as a part of the autodiscrimination process, whether decoding is to be continuous or discontinuous, etc. As will be explained more fully later, permitted combinations of scanning and decoding parameters together define the scanning-decoding relationships or modes which the reader will use.

[00069] After exiting block 627, the processor is directed to block 630 which, if the decoding attempt was not successful, is directed back to block 625 unless the trigger has been released (block 635) or unless reprogramming request has been received (block 640), or unless a stop or no-repeat request is called for by the current operating mode of the reader (block 642). The loop defined by blocks 625-642 will be the path repeatedly followed by the processor when autodiscrimination sequences are performed unsuccessfully, and no menuing or programming changes are called for, and no stop request is in effect. If this loop is interrupted by the user's release of the trigger, or by a successful decode, or by a reprogram request, or by a stop request, the reader will be directed by block 635 to stop and wait in a low power state until further processing is called for.

[00070] In the above-described loop, block 642 serves the function of stopping the repetitive scanning and decoding of the target symbol in those scanning-decoding modes or under those conditions in which a repetition of scanning and/or decoding is not called for. In the One Shot mode, for example, scanning and decoding are discontinued after one decoding attempt, whether or not that attempt is successful, without regard to the state of the trigger. Similarly, in the Repeat Until Stopped mode, scanning and decoding may be discontinued either by command, via block 642, or by the release of the trigger via block 635. Thus, block 642 comprises at least a part of the means by which the reader gives effect to the scanning-decoding parameters of the parameter table.

[00071] If block 630 indicates that the last decoding attempt was successful, the processor is directed to a block 645 which calls for a determination of whether the result of the decoding indicates that the decoded symbol was or was not a menu symbol. This determination may be made on the basis of results of the decoding, because all menu symbols are encoded with data that identifies them as such during decoding. If the decoded symbol is not a menu symbol, it is known that the symbol contained data that is to be output by the reader. In the latter event, the processor is directed to block 646, which causes it to output the data and, proceed to block 647.

[00072] Block 647, like block 642, comprises part of the means by which the reader gives effect to the scanning-decoding modes called for by the parameter table. In particular, if

decoding is successful (block 630) and has been output (block 646), block 647 discontinues scanning and decoding if the Repeat Until Done mode is in effect. If any other mode is in effect, scanning and decoding will continue unless blocks 635, 640 or 642 call for a different result.

[00073] If the decoded symbol is a menu symbol, block 645 directs the processor to perform the menuing routine called for by block 660 before returning to block 635. As will be explained more fully later in connection with Fig. 8, the latter routine enables the user to command the reader to perform any of a variety of different tasks, several of which include making user specified changes to the parameter table, thereby changing the operating mode of the reader, and the performance of any of a variety of user specified vector processing routines that do not change the parameter table. Once either of the latter tasks has been performed, the reader is directed to block 635, which causes it to capture and attempt to decode another image, in accordance with the parameters indicated by the parameter table, unless instructed to the contrary by blocks 635, 640 or 642. Optionally, the execution of menu routine 660 may be followed by a direction back to block 647, as indicated by dotted line 648, and the resultant discontinuation of scanning and decoding, if the reader is in its Repeat Until Done mode.

[00074] While reprogramming request block 640 has been described as being located between blocks 635 and 625, it actually preferably represents an externally generated interrupt request that may occur at any time that the reader is operating. Such a request may, for example, be initiated by a local host processor via one of I/O devices 37, 37' or 37". It may also be initiated by a remotely located processor, via one of the latter I/O devices, through a suitable transmission line or computer network, as shown in Fig. 9. However the reprogramming request is initiated, it directs the reader to execute the reprogramming routine called for by block 670. As will be explained more fully in connection with Fig. 10A, this routine causes the reader to be reprogrammed, either in whole or in part, thereby changing or updating the manner in which it operates and/or the symbols which it attempts to decode.

Menuing

[00075] The menuing feature will now be described with reference to Figs. 7A-7C, and the menuing flow chart shown in Fig. 8.

[00076] Turning first to Fig. 7A, there is shown the format for a menu message or word 650 of the type used. This menu word will ordinarily be produced as a result of the decoding of a menu symbol, selected by the user, from a collection of menu symbols printed in a User's Manual supplied with the reader, along with a description of their functions.

[00077] Menu word 650 begins with a first one-byte product identification (ID) code field 650-1 that identifies the type and/or model number of the reader. If the decoded product ID code indicates that it is compatible with the menuing program, execution of the menuing program continues normally. If it is not, the processor is caused to exit the menuing routine without making any menu specified changes.

[00078] The next field 650-2 of menu word 650 specifies the op code thereof in terms of a number from 0 to 7. This field specifies the operation to be performed by the menu word. The meanings of these different op codes are listed in Fig. 7C. Among these is op code "0," an op code that specifies some task that does not involve a direct change to the parameter table. Such operations will hereinafter be referred to as "vector processing operations". Exemplary ones of the tasks that may be requested pursuant to op code 0 are listed under headings A1-A4 of Fig. 7C, which tasks may be specified and differentiated from one another by the data included in the data fields 650-3 through 650-7 which follow op code field 650-2.

[00079] Specifically, the vector processing operations comprise selectable menu routines. Vectors to these routines can be stored in a vector table. The contents of data field 650-3, “offset,” is an index to the vector table relative to the base address thereof. If the offset field includes 10 bits, and only five of these bits are used as an index, then 32 different vector values will be possible. In this case the remaining 5 bits may be used for data.

[00080] The vector processing operations are preferably made selectable to a user by including respective menu bar code symbols in tables in the User’s Manual of the reader. The user may then select the desired vector routine by imaging the appropriate symbol. The manner in which such a table is used will be described later in connection with Figs. 8A-8D.

[00081] Among the vector processing operations which may be selected under op code 0 are the following. Operation A1 calls for the reader to output, *i.e.*, display or print, via the local host processor, or via an on-reader LCD display, the identity of the version of the software currently being used by the reader. Operation A2 calls for the reader to output the

current contents of the parameter table. Operation A3 calls for the reader to output the code options that are enabled, *e.g.*, the types of symbols that the reader is to attempt to decode during the autodiscrimination process and whether or not a "multiple symbols option" has been enabled. Other options may also be defined as desired.

[00082] Operation A4 is a particularly powerful and desirable vector processing operation which causes the printer of the local host processor to print a menu bar code symbol that contains all of the information necessary to instruct another reader how it must be programmed if it is to operate in the same manner as the current reader. This, in turn, enables the user to quickly set up the same (or another) reader to operate in a manner that would otherwise require the user to manually select an entire sequence of parameter table values. If it is used to set up other readers, the process of using such a menuing bar code symbol may be thought of as a "cloning" procedure, since it allows a multiplicity of readers to be identically configured.

[00083] The type of bar code symbol in which the parameter table is printed must naturally be in a bar code symbology in which the reader is able to both encode (or write) data and decode (or read) data. Because the parameter table has a data content which may be too high to be encoded in many 1D symbologies, the menu symbol encoding the parameter table is preferably encoded in a 2D bar code symbol. One 2D symbology which is particularly suitable for use in encoding a menu bar code symbol of the subject type is that developed by Welch Allyn, Inc. and referred to as the "Aztec" symbology. The manner in which data is encoded in accordance with the Aztec symbology is described in detail in copending, commonly assigned U.S. Patent No. 5,591,956, which is hereby expressly incorporated herein by reference.

[00084] In addition to op code 0, menu word 650 also makes available op codes 1-7, as shown in Fig. 7C. The latter op codes comprise simple commands, each of which specifies a change that is to be made at a particular part of the parameter table, using specified data, if required. Assuming that parameter values are stored as bytes in respective addresses of the memory that are set aside for use as a parameter table, offset field 650-3 will comprise an index to the parameter byte relative to the base address of the table. The data or data mask that is to be used with the specified offset is specified by the data contained in up to four 8 bit data fields 650-4 through 650-7 of menu word 650.

[00085] Referring to Fig. 7C, for example, op code "1" specifies a "clear" operation. It directs the processor to the byte of the parameter table that is pointed to by the offset field, and uses the content of data field 650-4, Data 0, to specify the bit mask that is to be used to specify the bits to be cleared. Op code "6", on the other hand, specifies a load operation. It directs the processor to the byte of the parameter table that is pointed to by the offset field, uses Data 0 as the bit mask for the bits to be changed, and uses Data 1 as the new data for those bits. Because the use of op codes of this type are known to those skilled in the art, the use of these op codes will not be described in detail herein.

[00086] The parameter table is used to specify the operating options that are made subject to the control of the user. Representative groups of such options are shown as headings A-E of Fig. 7B, together with some of the options that may be selected under those headings. One important group of those options are those that are labeled as "code options" under heading B. Under this heading may be found the parameter table addresses that are set aside for use in specifying the enabled/disabled states of the various decoding programs that may be used during the autodiscrimination process. The parameter table addresses corresponding to options B1 and B2, for example, may be set aside for specifying whether all 1D codes or all 2D codes are or are not to be used in an attempt to decode an unknown symbol during autodiscrimination. Similarly, the parameter table address corresponding to option B3, may specify a particular bar code symbology, such as MaxiCode, that is to be enabled or disabled, *i.e.,* specify whether the autodiscrimination process is or is not to include an attempt to find a MaxiCode symbol in an image. In addition, the parameter table address corresponding to option B4 may indicate that after decoding, messages that are longer than a specified maximum length or shorter than a specified minimum length are not to be output. Depending on the application, this Min-Max length option may be applied on a symbology dependent basis, *i.e.,* applied so that it is active with some symbologies, but not with others, or may be applied on a symbology independent basis. Finally, the parameter table address corresponding to option B5 specifies whether the Multiple Symbols option is or is not to be used. The enablement of this option, which given effect by block 643 of Fig. 6A, calls for the reader to attempt to decode more than one symbol in the field of view of the reader without having to acquire multiple images of that field of view. The types of options selected for inclusion under heading B will vary from application to application, and will be understood not to be restricted to any particular selection of such types.

[00087] The inclusion of user selectable code options as part of the menuing process has a significant effect on the overall data throughput rate of the reader, *i.e.,* on the time necessary to decode a symbol whose symbology is not known in advance. If, for example, it is known that none of the symbols to be read during a series of readings comprise 1D symbols of any type, or any subset of 1D symbols such as Codabar, Code 39 or Code 128, code options allow a user to direct that any attempt to decode an unknown symbology according to these symbologies is to be skipped, thereby shortening the total time necessary for the processor to decode the unknown symbol according to the symbology which it does use. This skipping also reduces the chances of a misread. If, on the other hand, it is known that all of the symbols to be read during a series of reading operations are of one type, such as Interleaved 2 of 5, all 2D decoding programs and all the decoding programs for 1D symbologies other than interleaved 2 of 5 may be disabled, thereby limiting all decoding attempts to a single 1D symbology. Thus, the menuing process allows the autodiscrimination process to be optimized so as to achieve the highest possible data throughput rate.

[00088] A second important group of options provided by the menuing process are those that are labeled as "Scanning-Decoding" Options under heading C of Fig. 7B. Unlike the code options of heading B, the scanning-decoding options of heading C are not concerned with which codes are enabled or disabled, but rather with the relationships which will be allowed to exist between scanning and decoding. The parameter table address corresponding to option C1, for example, may be used to specify that the reader operate in a "One Shot" scanning-decoding mode. In this "One Shot" mode the reader will scan and attempt to decode one bar code symbol each time that the trigger is depressed and then stop. The address spaces corresponding to scanning-decoding modes C2 and C3, on the other hand, may be used to specify that the reader operate in a "Repeat Until Done" (RUD) or "Repeat Until Stopped" (RUS) scanning-decoding mode. In these modes, the reader will scan repeatedly and attempt to decode repeatedly until there is a successful decode (RUD), or until requested to stop whether or not there is a successful decode (RUS). Scanning-decoding modes C1 - C3 are preferably made user selectable by including suitable menu symbols in the User's Manual.

[00089] Also included among the scanning-decoding modes are the tracking modes listed under headings C4 - C6 of Fig. 7B. Of these, the Scan On Demand (SOD) mode C4, when enabled, causes decoding to proceed continuously while scanning is started and stopped as

necessary to maintain a tracking relationship between scanning and decoding. Skip Scan (SS) scanning-decoding mode C5, when enabled, causes the results of older scans to be discarded in favor of more current scans when and as necessary to maintain the desired tracking relationship between scanning and decoding operations. Finally, Decode On Demand (DOD) scanning-decoding mode C6, when enabled, causes scanning to proceed continuously while decoding is started or stopped as necessary to maintain a tracking relationship between scanning and decoding. The particular one of these tracking modes that will be used is preferably set during manufacture, based on the amount of image data memory that is present within the reader, and not changed thereafter. There is no reason in principle, however, why tracking options C4 - C6 cannot be made user selectable as, for example, by the inclusion of suitable menu symbols in the User's Manual.

[00090] The availability of the SOD, SS and DOD tracking modes among the scanning-decoding options that may be selected during the factory programming of the reader is beneficial since it allows the data throughput rate of the reader to be optimized in view of the amount of memory that is available within the reader. At the same time, because operation in all of these modes may be disabled during operation in the One Shot, Repeat Until Done, or Repeat Until Stopped modes, the reader is able to operate in accordance with the non-tracking variants of these modes when such operation is preferred. One condition under which such operation may be preferred is one in which scanning while decoding is slow as a result of the time sharing of a bus. Thus, the reader combines flexibility of use with time-optimized use of the scanning and memory resources of the reader.

[00091] As will be explained more fully later, the RUD and RUS modes may be used either with or without one of the above-described tracking modes. This is because repetition is a necessary but not a sufficient precondition to the use of the tracking modes. Accordingly, if the RUD or RUS mode is not used in conjunction with a tracking mode it will comprise a non-tracking mode. If the RUD or RUS mode is used in conjunction with a tracking mode it will comprise a tracking mode.

[00092] Other groups of options that are provided by the menuing feature include those that are set aside under headings A, D and E of Fig. 7B. Of these Communication Options, heading A, is associated with parameter table addresses that correspond to various communication modes that may be used by the reader. Included among these options are A1,

an option that enables/disables RS-232 communication through an I/O device (such as I/O 37, 37′, etc.), A2 which specifies the baud rate of the selected communications mode, and A3 which enables/disables the RF link that the reader may use in place of multi-conductor cable 54-2 of Figs. 4A - 4C. Option A4 is an example of a network option which specifies the type of computer network with which the reader is to operate, in this case ETHERNET, although other types may also be provided for.

[00093] Similarly, heading D is associated with parameter table addresses that correspond to various miscellaneous operating options that may be selected by the user. Included among these options are D1 which enables/disables the beeper and allows the volume thereof to be adjusted, D2 which enables/ disables the use of an aiming LED, and D3 which enables/disables the provision of aural feedback to the user, among others. An example of a reader which provides aural feedback is described in U.S. Patent No. 5,420,409.

[00094] Heading E is associated with parameter table addresses that correspond to various transmission options that may be selected by the user. Included among these options are E1 and E2, which enable/ disable the outputting of check characters or checksum data with decoded data, and E3 which enable data edit options such as adding a carriage return and/or a line feed and/or other ASCII characters to the decoding data. Options E1 and E2 are useful, for example, in the localization and identification of hardware or software failures during the servicing of the reader. Option E3 is useful in placing decoded data in a form suitable for use with an application program.

[00095] Heading F is associated with parameter table addresses that correspond to various message editing commands for editing the form of characters in a decoded message. These commands may be, for example, search and replace commands (option F1), commands to insert characters (option F2), commands to delete characters from a decoded message (option F3), or other commands.

[00096] Heading G, meanwhile, is associated with parameter table addresses that correspond to commands for adding prefixes or suffixes, of a selectable character length, to a decoded message. Prefixes and suffixes are added to messages so that the host processor can identify the source of, or other characteristics of received messages. Option G1 allows addition of a prefix to a decoded message while option G2 allows addition of a suffix to a decoded message.

[00097] In view of the foregoing, it will be seen that the menuing process provides a wide range of user selectable functions and modes that allow the reader to be tailored to a user's specific application and/or preferences. Among these, the code options and the scanning-decoding options in particular, allow a user to reconfigure the operation of the reader in ways that have not heretofore been possible and thereby substantially increase the flexibility and overall data throughput rate of readers.

[00098] The manner in which the reader can be updated to accomplish the above-described results will now be described with reference to the flow chart of Fig. 8, which shows the steps included within menu routine block 660 of Fig. 6A. The menu routine of Fig. 8 begins with a block 805 which causes the processor to convert the decoded menu symbol message into hexadecimal form. This has the effect of formatting the message so that the fields of the menu word are expressed as pairs of hexadecimal digits. Once this has been done the processor examines the product ID code to verify that it is compatible with the reader being menued. If it is not, the processor is directed to exit the menuing routine and continue scanning. If it is, the processor is directed to block 810 which distinguishes those menu messages which contain op codes from those which contain numerical data but no op codes. If there is no op code, the processor is directed to block 815, which causes it to collect in an accumulator all of the digits of the message for later use before proceeding to block 850. An example of numerical data without an op code comprises the minimum or maximum length of the messages that are to be output under code option B4.

[00099] If the menu message contains an op code, and the op code is other than 0, the processor is directed, via block 820, to a block 825. The latter block causes it to make the parameter table changes called for by the op code and the associated offset and data fields, sets a "flash" flag to indicate that changes have been made and then proceeds to block 850. This has the effect of implementing the user selected changes in the menuing options discussed previously in connection with Fig. 7B. Such changes will ordinarily be made in a copy of the parameter table that is stored in RAM 45, and then later transferred to EROM 46.

[000100] If the menu message contains an op code of 0, the processor is directed, via block 820, to a block 830. The latter block causes the processor to perform the vector processing operation indicated by the remainder of the message. This operation will comprise one of the

operations discussed previously in connection with items A1-A4 of Fig. 7C, among others, before proceeding to block 850.

[000101] In view of the foregoing, it will be seen that, when the processor arrives at block 850 it will have taken all required numerical data, performed all required parameter table modifications, or performed all required vector processing operations. As will now be explained, the remainder of the flow chart of Fig. 8 is directed to storing a semi-permanent copy of the parameter table in EROM 46.

[000102] If, on arriving at block 850, the processor finds that the "flash" flag has not been set, it knows that the contents of the parameter table have not been changed and, consequently, that no updated copy thereof needs to be stored in EROM 46. Under this condition, the processor is directed to simply return to the main program of Fig. 6A. If, on arriving at block 850, the processor finds that the "flash" flag has been set, however, it knows that the contents of the parameter table have been changed and, consequently, that an updated copy thereof needs to be stored in EROM 46. Under this condition, the processor is directed to blocks 855, 860 and 865, which defines the steps necessary to store this updated copy.

[000103] In accordance with block 855, the processor is instructed to copy from EROM 46 to RAM 45, the program instructions (flash routine) necessary to copy the parameter table from RAM to EROM. The copying of the flash routine to RAM is necessary because the EROM cannot be written to when the apparatus is reading or operating from the EROM. Once the flash routine has been copied to RAM 45, the processor is directed to jump to RAM to begin executing that routine. As it does so it is directed, via block 860, to erase the old (unchanged) parameter table from EROM 46. Per block 865, it then copies new (changed) parameter table from RAM 45 to EROM 46. Once this has been done, the processor is directed back to the main program of Fig. 6A to begin operating in accordance with the operating mode specified by its new parameter table. Thus, the performance of the steps called for by blocks 855-865, when called for by block 850, has the effect of partially reprogramming the reader so that it operates in the manner indicated by the last menuing symbols selected by the user.

[000104] Referring to Figs. 8A-8D, there are shown examples of menu symbol selection charts of the type that may be used. Referring first to Fig. 8A, there are shown two parts of an option selection or menu chart that is used to enable and disable two exemplary 1D bar code symbologies, namely: Code 128 and UPC A. If a user wants to enable the decoding of Code 128 symbols, he need only image menu symbol 802 which, in the present example, is a 2D bar code symbol expressed in the Aztec bar code symbology. Conversely, if a user wants to disable the decoding of Code 128 symbols, he need only image menu symbol 804. Similarly, imaging symbols 806 or 808 enables or disables the decoding of UPC A symbols. Advantageously, the change called for by the user is accomplished as the result of a single imaging step, rather than as a result of multiple imaging steps.

[000105] Referring to Fig. 8B, there are shown two parts of an option selection chart that is used to select the desired one of the baud rates that may be used by the reader's I/O devices. A user chooses the desired one of the exemplary 1200, 9600, 19200 and 38400 baud rates by simply imaging the corresponding ones of menu symbols 812-818. Again, the change is accomplished as the result of a single imaging step.

[000106] The fact that the above-discussed examples of menu selections make use of menu symbols that use the Aztec 2D symbology is not essential to the practice. Other 2D or 1D menu symbol symbologies could also have been used, if desired, as will be seen from the following discussion of Figs. 8C and 8D. What is important is that the symbology used for the menu symbols be the one that is correct for the model indicated by the product ID field of the menu word. In the case of Figs. 8A and 8B, the illustrated menu symbol symbology is that which is used by the IMAGETEAM™ Model 4400 reader manufactured by Welch Allyn, Inc.

[000107] Referring to Fig. 8C, there are shown exemplary parts of the option selection or menu chart that can be used with Welch Allyn SCANTEAM® readers. In Fig. 8C, symbol 822 is an example of a menu symbol that, if imaged, causes all Code 11 and Code 128 settings to assume their default values. Symbols 824 to 836 are examples of menu symbols that allow Code 11 options to be enabled and disabled on an individual basis. Similarly, symbols 848 to 856 are examples of menu symbols that allow Code 128 options to be enabled and disabled on an individual basis.

[000108] Referring to Fig. 8D, there are shown further exemplary parts of the option selection or menu chart that may also be used with Welch Allyn SCANTEAM® readers. In Fig. 8D symbol 858 is an example of a menu symbol that, if imaged, causes the settings for one of the RS-232 ports of the reader to assume their default values. Symbols 862 and 864 are examples of menu symbols that enable and disable a CTS check selection feature. Finally, symbols 866 through 884 are examples of menu symbols that allow any of a number of different baud rate selections to be made. Once again, the present reader allows all of these menu selections to be made by means of a single step selection process.

[000109] Because fuller information concerning the menu options contemplated, and their uses is contained in the User's Manual for the above-identified readers, these menu options will not be discussed further herein.

Reprogramming

[000110] In accordance with another feature of the apparatus and method, the reader may be reprogrammed to operate in accordance with an entirely new application program. This means that the reader may not only be provided with a new or updated decoding program, or a new parameter table, it may also be provided with one or both of a new menuing routine and a new main program. As a result, a reader may be effectively reconfigured as a new reader, with new capabilities and features, as often as necessary to keep it up to date with the latest developments in optical reader technology. Advantageously, this reprogramming may be accomplished either locally as, for example, by a local host processor equipped with a diskette or CD-ROM drive, or remotely by a distant processor that is coupled to the reader via a suitable telephone or other transmission line or via a computer network or bulletin board.

[000111] The reprogramming feature will now be described with reference to the system block diagram of Fig. 9 and the reprogramming flow chart of Fig. 10A. Referring first to Fig. 9 there is shown a reader 10, of the type shown in Fig. 4 or 5, which is coupled to a local host processor 900 by means of multi-conductor flexible cable 54. The reader may also comprise a cordless battery powered reader 10' which is coupled to a host processor 900 via a suitable RF link including antennae 905 and 910 and an RF interface module 915. Host processor 900 is preferably equipped with a display 930 by which the results of the previously described vector processing operations may be displayed, and with a printer 940 by which the previously described menuing bar code symbol may be printed. As used herein, the term "local host processor" will be understood to include both stand alone host processors and host processors which comprise only one part of a local computer system.

[000112] If the new reader program is available locally as, for example, on a diskette or CD-ROM, it may be loaded into reader 10 or 10′ using a suitable drive unit 920, under the control of a keyboard 925 and the reprogramming routine shown in Figs. 10A and 10B. In addition to drive unit 920, processor is typically in communication with a read only program storage device such as a ROM 921 and a read/write storage device such as a RAM 922. If the new reader program is available at a remotely located processor 950, it may be loaded into reader 10 or 10′ through a suitable transmission link 955, such an electrical conductor link, a fiber optic link, or a wireless transmission link through a suitable communication interface 960, such a modem. As used herein, the term "transmission link" will be understood to refer broadly to any type of transmission facility, including an RS-232 capable telephone line, as called for by communication option A1 of Fig. 7B, an RF link, as called for by communication option A3 of Fig. 7B, or a computer network, *e.g.,* ETHERNET, as called for by communication option A4 of Fig. 7B, although other types of transmission links or networks may also be used. For example, transmission link 955 could be provided by a coaxial cable or any other non-RF electromagnetic energy communication link including a light energy infrared or microwave communication link. Link 955 could also be an acoustic communications link. Additional communication options include a baud rate option A2 which allows different baud rates to be selected.

[000113] The manner in which the reader may be made to perform any of a variety of different externally specified functions, including reprogramming itself, will now be described with reference to the flow charts of Figs. 10A and 10B. As will be explained more fully presently, the flow chart of Fig. 10A is a flow chart by which a program originating outside of the reader may be loaded into the reader for execution thereby. One example of such an externally originated program is the reprogramming program shown in Fig. 10B. Other examples of such externally originated programs may include diagnostic or test programs, among others.

[000114] Turning first to Fig. 10A, this flow chart is entered when the reader receives an externally generated command, such as the six character sequence BBOOTT, which it is programmed to recognize and respond to. This command may be initiated either by a local or a remotely located processor and transmitted to the reader via any of the I/O devices shown in Fig. 1. It may, for example, be initiated by the local host processor via keyboard 945 or by remote processor 950. This command may be given effect as an interrupt request and

recognized as such by decision block 1005 of Fig. 10A. It will be understood that while interrupt block 1005 is shown in Fig. 10A, it may in fact be located at any point within the main program of the reader.

[000115] Once the BBOOTT command has been received and acted on, the processor enters a loading loop including blocks 1007 through 1020. This loading loop causes the processor to load a program into RAM, one line at a time, in conformity with any suitable communication protocol, until the last line of code is detected via block 1020. When the latter has occurred, the processor is directed to block 1025, which causes it to jump to the newly received program and to begin executing the same before returning to the main program.

[000116] Referring to Fig. 10B, there is shown an exemplary flow chart for a reprogramming routine suitable for use in reprogramming the reader to operate with new or different decoding programs, and or new or different menuing programs, among others. This program is an example of a program which may be executed as a result of the execution of the loading loop 1007-1020 of Fig. 10A, and which begins to be executed as the processor enters block 1025 of Fig. 10A.

[000117] On executing the reprogramming flow chart of Fig. 10B, the device loads the program that is intended to replace all or part of the program currently stored in EROM. This process begins as the processor encounters block 1035, which directs it to wait until a line of externally generated code is received. As each line of code is received, it is first checked for correctness (e.g. checksum), as called for by block 1040 and, if an error is found, sends a negative acknowledgment signal to the sending processor per block 1045. Each time that a correct line of code is received, the flow loops back for additional lines until the last line of the current file has been correctly read, as called for by block 1050. Since the last line of the file does not contain program data, and cannot occur until all blocks of program data have been processed, block 1050 will direct the processor to block 1060, unless and until all blocks of program data have been received and stored in EROM 46, and then cause it to return to the main program of Fig. 6A via exit block 1055.

[000118] If the processor has not exited the reprogramming routine of Fig. 10B per blocks 1050 and 1055, block 1060 will cause it to determine if the last received line indicated that a new block has begun. If it has, the processor is directed to block 1065, which causes it to erase that new block of EROM before continuing to block 1070 and storing that last received line therein. If it has not, block 1070 will cause the processor to store the last received line to the last erased block of EROM. If this line has been successfully stored, as determined by block 1075, the processor will acknowledge that fact per block 1077 and loop back for another line.

[000119] If, however, any line of data has not been successfully stored, block 1075 will direct the processor to a block 1080 which causes it to output an error message and exit the program. If the latter occurs, the reprogramming routine as a whole must be repeated. If the latter does not occur, the above-described process will continue line-after-line, block-after-block, until the entire file has been successfully transferred.

[000120] In view of the foregoing, it will be seen that the effect of the reprogramming routine of Fig. 10B is to attempt to reprogram part or all of EROM 46 as requested, or to continuing the attempt to do so until it either succeeds or fails. To the extent that the reader is reprogrammed, it will effectively become a new or updated reader. This is not only because this reprogramming can not only modify the parameter table, it can also modify the decoding or other programs referenced by the parameter table and the menuing program itself. Thus, the reprogramming feature can not only change the manner in which the reader operates, it can also change the manner in which the operation of the reader can be modified in the future.

[000121] With the use of the above-described reprogramming feature, the reader may be kept current with the latest available programs that are suitable for use therewith. A user at local processor 900 may, for example, communicate with remote processor 950, via keyboard 925, and determine if new programmable features are available. If they are, he may obtain them from the remote process and download them locally, or request that the remote processor download them directly to the reader. Alternatively, the remote processor may initiate the reprogramming of the reader independently as, for example, pursuant to a service contract or updating service. It will be understood that all such embodiments are within contemplation.

Local Host and Reader System Operations

[000122] As has been described hereinabove, reprogramming of a reader may be accomplished with use of a local host processor. This section describes additional features of a system comprising a local host processor 900 and a reader 10, and more particularly describes features and additional system operations that are realized by various interactions between host processor 900 and reader 10, and in certain applications by a host processor 900 that is not in communication with a reader 10.

[000123] A flow diagram illustrating the primary program for operating a local host processor for use in controlling a reader is shown in Fig. 11A. By executing block 1102 host processor causes to be displayed on a display monitor 930 a subprogram option screen. The subprogram option screen displays various subprogram options for a user to select. Selection of one subprogram option causes a series of instructions pertaining to that particular option to be executed by local host processor 900. These subprograms of a host primary program controlling local host processor may include, for example, a subprogram for reprogramming a reader; a subprogram for uploading parameter information from a reader to host, or information pertaining to a main program presently operating a reader; a subprogram for instructing a reader to perform self-diagnostic testing; a subprogram for determining the reader's main program revision level; a subprogram for outputting parameter table information, possibly to auxiliary readers; a subprogram for editing parameters of a parameter table; a subprogram for simulating the result of applying editing commands to a decoded message; and a subprogram for displaying barcode symbols for scanning by a reader.

[000124] A flow diagram illustrating a subprogram for reprogramming of a reader 10 by control of a local host processor is shown in Fig. 11B. Whereas Figs. 10A and 10B illustrate instructions executed by processor 40 of reader 10 for providing reprogramming of a reader, Fig. 11B illustrates instructions executed by local host processor for providing reprogramming of a reader.

[000125] At block 1110 host processor 900 displays a main reprogramming screen on display monitor 930. The main reprogramming screen prompts a user to designate a source for an operating program. The source designated is typically a bulk storage device such as a hard or floppy disk drive but also may be, for example, a RAM or ROM storage device.

When the source is selected, host processor 900 displays on display monitor 930 indicators of the operating programs, or files, that are available in the storage device source selected (block 1114) and a user selects one of the operating programs. Some available operating programs comprise entire main programs and entire parameter tables for loading into reader, whereas other available operating programs include only parameter tables which may be customized parameter tables created by a user during execution of a parameter table editing subprogram.

[000126] When a user selects a source for an operating program, and selects a desired operating program, downloading of the operating program proceeds. At block 1116 host processor determines whether a reader is connected to the host processor communications link, normally by serially transmitting a device detection command to a reader, which has been previously programmed to transmit an acknowledge response message on the reception of a detection command.

[000127] If a reader is connected to host processor 900 then host processor at block 1118 sends an identification command to reader 10 which is previously programmed to transmit an identification response on the reception of an identification command. After receiving the identification response and comparing the response to the selected operating program at block 1120 processor at block 1122 determines whether the reader is of a type which is compatible with the selected operating program. An operating program is compatible with a reader in communication with host processor if the operating program is specifically adapted for that reader's unique hardware configuration. Bar code readers of various types have different hardware components including different memory devices, image sensors, input/output devices, and other components. The selected operating program must be in form enabling it to communicate with the particular hardware components of the presently connected reader.

[000128] If the selected operating program is compatible with the present reader, the host processor at block 1126 determines if the operating program is a parameter-only type operating program or an operating program that comprises a main program and a parameter table. This determination can be made, for example, by reading the contents of a DOC type file which is made to be read by processor 900 when an operating program is read, and which is made to include an identifier as to whether the operating program is of a type which includes a main program and parameter table; by reading the contents of a predetermined address of the operating program which is made to include an identifier as to the type of operating program; or by reading predetermined addresses of an operating program designated for storing a main program and basing the determination on whether instructions are present in the designate addresses.

[000129] A memory map for a typical operating program is shown in Fig. 11C. When an operating program is stored in a memory device, which may be, for example EROM 46 of reader 10, or a disk drive 920 or other storage device associated with host processor 900 a plurality of first predetermined address locations e.g. 000 to 5000 of the storage device are designated for storing parameters of the main program, while a plurality of second predetermined address locations e.g. 8000 to 9000 are designated for storing instructions of a parameter table. The beginning and end addresses of the parameter table may change from operating program to operating program. However, the parameters of each parameter table are in identical locations with respect to the beginning address.

[000130] When host processor 900 determines at step 1126 that the selected operating program includes a main program then program control proceeds to step 1130 wherein processor transmits the contents of the selected operating program into EROM 46 of reader 10. If host processor 900 determines at block 1126 that the selected operating program is a parameter only type operating program then host processor 900 first queries EROM 46 to determine the begin and end address locations of the parameter table of the operating program currently stored in EROM. To this end host processor 900 at block 1130 polls the contents of a vector pointer table 1134 in predetermined address locations of EROM. Described previously herein vector pointer table 1134 comprises the beginning and end addresses of the parameter table. After vector pointer table is polled, host processor 900 stores the address location of the present parameter table, modifies the parameter table address of the selected parameter-only operating table in accordance with the parameter table addresses of the existing parameter table (block 1136) and writes the contents of the parameter table address locations of the modified parameter-only type operating program into EROM 46 (block 1140).

[000131] If processor 900 determines at block 1126 that the selected operating program is of the type having a main program and a parameter table, then processor 900 at block 1144 prompts the user whether the user would like to save the contents of a parameter table of the operating program currently stored in EROM 46 of reader 10; that is, utilize the parameters of

the current operating program in the operation of a reader that is programmed to have a new main program. If the user responds affirmatively, then processor 900 reads the contents of the existing parameter table (block 1150) after first polling the vector pointer table and then writes, at block 1152, the contents of the existing parameter table in a predetermined holding address location of a storage device associated with processor 900 or reader 10.

[000132] The selected operating table is then written into EROM 46 at block 1140, line by line, until loading is complete. If the user had requested at block 1144 to save the contents of the original parameter table (a determination made at block 1153), then processor 900 writes the contents of the parameter table stored in a holding address location to the appropriate parameter address locations of EROM at block 1154, after determining the address locations of the parameter table at block 1156. Referring again to the primary host processor program shown in Fig. 11A, another subprogram which can be selected from subprogram option screen displayed at block 1102 is a subprogram for editing a parameter table via host processor control. An important feature available in this subprogram is that the subprogram allows a user to edit a parameter table read from a memory location of processor 900 or reader 10 without there being a reader currently in communication with processor 900, thus improving the convenience of operation.

[000133] As discussed previously with reference to Fig. 7B, a parameter table is used to specify operating options that are subject to the control of the user. During execution of instructions of a reader's main program stored in a first predetermined memory locations of a storage device, parameters of a parameter table, which is stored in a second predetermined set of memory address locations of a storage device, are called up with use of lookup type instruction as exemplified by representative lookup instruction (in pseudocode) 1160 shown in Fig. 11C. Parameters of a parameter table may be, for example, communications option parameters (subheading A), code option parameters (subheading B), scanning-decoding option parameters (subheading C), operating option parameters (subheading D), transmit option parameters (subheading E), data formatter command parameters (subheading F), prefix/suffix parameters (subheading G), or other types of parameters.

[000134] A flow diagram for a parameter table editing subprogram is shown with reference to Fig. 11D. At block 1162 processor 900 determines if a reader is in communication with processor 900 in the fashion described previously with reference to block 1116 of Fig. 11B.

If a reader is present, processor 900 at block 1166 reads the parameter table presently stored in EROM 46 (after determining the table's location), along with a list of analog waveform outputs from another predetermined memory location from EROM 46. A list of possible types of analog waveform outputs a reader may be configured to generate allowing the reader to transmit data to various types of terminals is stored in a predetermined waveform list memory location. The waveform list memory location may be determined by querying vector pointer table 1134. A specific one type of waveform output from the list of available outputs is selected by control of a parameter of parameter table, typically stored in an address location corresponding to Communications Options (Heading A) type parameters described previously with reference to Fig. 7B. Processor 900 at block 1116 stores the parameter table and the list of analog waveform outputs in a temporary storage device associated with processor 900 such as a RAM.

[000135] In the embodiment shown, the parameter table editing subprogram is configured by default to edit the existing parameter table stored in EROM of the connected reader if a reader is present. It will be recognized, however, that the editing subprogram can also be configured to query the user as to whether the user wishes to edit the parameter table currently stored in reader EROM 46, or another candidate parameter table typically stored in a bulk storage device associated with processor 900.

[000136] If a reader is not in communication with processor 900, continuing with reference to the flow diagram shown, then processor at block 1168 prompts the user to select a reader for which the user wishes to edit a parameter table and once a type of reader is selected, a default parameter table associated with that reader type is written in to a temporary storage device of processor 900 typically provided by a RAM device.

[000137] At the termination of block 1168 or block 1166 if a reader is connected, a parameter configuration screen is displayed to a user, at block 1169, an exemplary embodiment of which is shown in Fig. 11E. Typically, a user will edit certain parameters from the parameter table which the user wishes to change, and then, when editing is complete, a user will select an available output option from the parameter configuration screen. The output options available to a user may include writing an edited parameter table to a connected reader; writing an edited parameter table to a bulk storage device; displaying an edited parameter table; or printing an edited parameter table.

[000138] Until an output option is selected, the user typically edits various parameters the user wishes to change as shown in blocks 1170 and 1172. Selection of one parameter type option, e.g. code or symbology option parameter 1174 causes a secondary editing screen to appear allowing editing of parameters of the selected parameter type. When editing pertaining to one or several parameter types is complete then program reverts back to parameter configuration screen at block 1169, allowing user to select an output option.

[000139] If a user selects the write output option (block 1176), the edited parameter table is written to, or downloaded to reader EROM in the fashion described previously with reference to block 1140 of Fig. 11B. If a user selects the store-to-disc option (block 1178) then the edited parameter table is written to an address location of a bulk storage device such as a hard drive or floppy disc. If a user selects the display option (block 1180) then processor 900 causes the complete or partial contents of the edited parameter table to be printed on display screen associated with host processor 900. If a user selects the print option (block 1182) then processor 900 causes the complete or partial contents of the edited parameter table to be printed by a printer device 940 in communication with processor 900.

[000140] Another output option available to a user is to compare two or more parameter tables. If this option is selected (block 1184) then the user is requested at block 1186 to select parameter tables from memory locations (which may be memory location associated with processor 900 or with reader 10). When parameter tables have been selected, processor 900 at block 1186 compares the selected parameter tables. In general, the comparison is carried out by a compare function applied after an offset between the files is accounted for. Processor 900 then outputs the results of the comparison at block 1188, typically by displaying the comparison results on screen 930, or printing the comparison results using printer 940.

[000141] One specialized output option allows the user to create programming menu symbols whose general features have described with reference to Fig. 7A-7C, and 8. The menu symbols created by the output option can be used to reprogram readers reading the created symbols in accordance with the changes made to a parameter table made during execution of the parameter table subprogram. Described as a routine executed during a parameter table editing subprogram, the menu symbol output option can be conveniently implemented as a separate subprogram.

[000142] When a menu symbol output option is selected at block 1189, processor 900 determines at block 1202, by reading a reader identifier, whether the reader designated for receipt of the edited parameter table includes a one dimensional (1D) or two-dimensional (2D) image sensor.

[000143] If the reader includes a one dimensional image sensor, then processor 900 creates a series of linear bar codes which may be used for reprogramming several readers. Specifically, if the designated reader includes a one dimensional image sensor then processor 900 at block 1204 creates a first linear menu symbol adapted to generate an instruction causing the reader reading the symbol to change parameter table values of the reader's EROM to default values. Then, at block 1206 processor 900 creates a distinct linear programming menu symbol for each parameter of the parameter table that is changed during the editing process from a default value. An important feature is described with reference to block 1208. When the series of menu symbols is created, the created symbols may be printed on paper by printer 940 according to a conventional protocol, or else displayed on display device 930, typically a CRT monitor. The term created symbols herein refers to binary encoded data stored in a memory space which result in an actual symbol being output when the data is written to a display device or printer. An unlimited number of bar code readers may be reprogrammed by reading the menu symbols that are displayed on the display device 930. Displaying the created menu symbols on a display device allows rapid output of created symbols and eliminates the need to supply a paper substrate each time a menu symbol is output.

[000144] If the reader designated for reprogramming includes a 2D image sensor, then processor 900 at block 1210 need only create one 2D menu symbol in order to cause reprogramming of the designated reader in accordance with the changes made to a parameter table even in the case where multiple changes to the parameter table are made. This is so because an increased number of instructions may be encoded in a symbol of a 2D symbology type.

[000145] Another subprogram which may be selected from a subprogram option screen displayed at block 1102 is a subprogram for simulating the result of applying editing commands to a decoded message. As discussed previously, editing commands may be applied to decoded messages by entry of the commands to a parameter table in parameter table addresses corresponding to heading H of Fig. 7B. Without an editing command simulation subprogram, it would be necessary to decode a symbol with use of reader 10 in order to observe the result of applying the editing commands. The efficiency and convenience advantages of the editing command simulation subprogram therefore should be clear to those skilled in the art.

[000146] An exemplary flow diagram for an editing command simulation subprogram is shown in Fig. 11E. At block 1214 processor 900 displays a message editing simulation screen or screens which allows a user to enter an unedited test message and symbology type (block 1216) and enter the type of editing command desired to be applied to the message (block 1218). Three basic types of editing commands are search and replace editing commands, insert character editing commands, and delete character editing commands. Additional, more complex editing commands may also be applied.

[000147] When the commands are entered, processor 900 applies the commands entered at block 1218 to the unedited test message at blocks 1220, 1222, and 1224 if all are applicable. When editing is complete processor 900 outputs the result of applying the editing commands, at block 1226, typically by displaying the edited message on display screen 930.

[000148] At block 1228 processor queries the user as to whether the user wishes to save the editing commands which resulted in the edited message being displayed or otherwise output at block 1226. If the user elects to save the editing commands, then processor 900 at block 1230 writes the commands to a predetermined command save memory location associated with processor 900. When the parameter table editing subprogram described with reference to Fig. 11D is later executed the commands saved in block 1230 of the message editing command subprogram may be read from the command save memory location during execution of block 1192 of the parameter table editing subprogram.

[000149] In addition to being adapted to download new or modified operating programs to reader 10 remotely, processor 900 can also be adapted to remotely transmit component control instructions to reader 10 which are executed by reader processor 40 substantially on receipt by reader 10 to control one or more components of reader 10 in a manner that can be perceived by a reader operator. For example, processor 900 and reader 10 can be arranged so that processor 900, on receipt of a command from a user, transmits a component control instruction to reader 10 which is executed by reader processor 40 to have the same effect as trigger 52 being manually pulled, or alternatively, being released. Instructions transmitted by processor 900 having the same effect as manually pulling and manually releasing trigger may be termed, respectively, "remote trigger activation" and "remote trigger release" instructions. Processor 900 and reader 10 can also be complementarily arranged so that, on receipt of a user activated command to remotely control reader 10, processor 900 transmits to reader 10 an instruction which is executed by reader 10 substantially on receipt of the instruction to turn on LED's 22 or to "flash" LED's according to a predetermined pattern, or to activate an acoustic output device such as speaker 38 to issue a "beep" or a series of beeps. Component control instructions for on-receipt execution which operate to control LED's 22 or speaker 38 are useful, for example, to signal an alarm condition, to indicate that a task is completed, or to attract the attention of a reader operator for any purpose.

[000150] Processor 900 and reader 10 can also be complementarily arranged so that, on receipt of a user activated command, processor 900 transmits to reader 10 a component control instruction which is executed by reader 10 substantially on receipt thereof to transmit data which is stored in memory 45 or in another memory device associated with reader 10 such as a long-term nonvolatile memory device. For example, a component control instruction received from processor 900 may be executed by reader 10 to upload from reader 10 to processor 900 image data that is stored in a specific memory location of reader memory 45 such as a reader memory location that stores the most recently captured image data captured by reader. Processor 900 may subsequently display such uploaded image data on display 930. Other component control instructions which may be transmitted from processor 900 to reader 10 for substantially on-receipt execution by reader processor 40 are instructions which, for example, cause predetermined indicia to be displayed by reader display 56, or which cause processor 40 to capture, by appropriate control over image sensor 32, a single frame of image data corresponding to the scene presently in the field of view of reader 10 in memory 45 or in another memory device.

[000151] It will be understood that certain component control instructions require that reader processor 40 execute a series of instruction steps, or repetitive instruction steps to cooperatively control more than one reader component. For example, a component control instruction commanding an optical reader to capture an image normally requires that processor 40 execute a series of instruction steps involving control of such components as LED's 22, components of the imaging assembly, and memory 45.

[000152] A modified reader operating program that adapts a reader to receive component control instructions from a remote local host processor for substantially on-receipt execution by reader 10 is shown in Fig. 6B. Reader 10 is readily enabled to receive and execute remote component control instructions by modification of the program loop indicated by block 605 of Fig. 6A wherein reader 10 waits in a low power state until a trigger is pulled. As shown by the flow diagram of Fig. 6B, block 605 may be modified to the form illustrated by block 605' so that reader executes block 610 and the ensuing blocks shown and described in connection with Fig. 6A in response either to a trigger being manually pulled or to the receipt of a remote trigger activation instruction from processor 900. Block 635 of the flow diagram of Fig. 6A may also be modified so that the reader is responsive either to a manual trigger release or to receipt of a remote trigger receive instruction. Reader 10 may also be made to exit the loop indicated by block 605' on the condition that another component control instruction for on-receipt execution by reader 10 is received. As is indicated by block 602 and block 603, reader 10 may be adapted to exit the loop indicated by block 605' and to appropriately control the component associated with the received instruction on the condition that a remote component control instruction is received from processor 900.

Scanning-Decoding/Autodiscrimination

[000153] The scanning-decoding and autodiscrimination features, and their relationships to the above-described menuing and reprogramming features, will now be described with reference to Figs. 6 and 12 - 18. More particularly, the combined operation of these features will be discussed in connection with Fig. 6A. The SOD, SS and DOD scanning-decoding modes will be discussed in connection with Figs. 13 and 14, and the OS, RUD and RUS scanning-decoding modes will be discussed in connection with Fig. 15. Finally, the 1D and 2D portions of the autodiscrimination feature will be discussed in connection with Figs. 16-18, respectively.

[000154] Turning first to the main program of Fig. 6A, the scanning and decoding operations are shown as blocks 625-647. In those embodiments or modes in which the multiple symbols code option is not enabled (see option B5 of Fig. 7B), the processor assumes, that only one symbol is to be decoded. Under this condition, if decoding is successful, the processor processes the decoded symbol as a menu symbol in accordance with previously described menu routine 660, or as output data in accordance with block 646, and then is stopped by one of blocks 647, 635 or 642. If decoding is not successful, the processor is directed back (unless stopped by blocks 635 or 642) to capture and attempt to decode another image. In this case, the "no" output of multiple symbols block 643 is selected, allowing additional images to be captured as necessary.

[000155] In those embodiments or modes in which the multiple symbols option is enabled, the processor assumes that more than one symbol is present in the image data. Under this condition, if decoding is successful, the processor continues to loop back to block 627 to make additional decoding attempts, unless stopped by one of blocks 635 or 642. In this case, however, the "yes" output of block 643 is selected, preventing additional images from being captured.

[000156] When the processor begins executing its scanning-decoding program, it first determines from the parameter table which scanning-decoding option or combination of options is to be used. It will then be directed to an autodiscrimination routine that is configured to execute that routine in accordance with the selected scanning-decoding option or options.

[000157] At start up, the parameter table maybe set up so that operation in the One Shot scanning-decoding mode is established as a default condition. Alternatively, the parameter table may be set up so that the RUD or RUS scanning-decoding mode is established as a default condition. Since the One Shot mode is inherently a non-tracking mode, its selection as a default mode implies that none of the tracking modes is selected. Since the RUD and RUS modes can be used either with or without one of the three tracking modes, its selection as a default parameter may or may not be associated with one of the three tracking modes, depending upon how the reader is programmed at the time of manufacture.

Tracking Options

[000158] The differences between the three tracking modes are best understood with reference to Figs. 12-14. The latter figures (with changes in figure and indicia number) are incorporated from prior copending U.S. Patent Application No. 08/914,883, now U. S. Patent No. 5,942,741, together with their associated descriptions as follows:

[000159] Scanning of indicia can take place under either of two generalized conditions, depending upon the decoding load presented by the indicia. Under light decoding loads,

shown in Fig. 12A for a prior art reader, the amount of data to be decoded is relatively small, allowing scan data from a complete scan to be decoded in a time which is less than the duration of a scan. Under this condition, the result of each scan is decoded before the completion of the following scan, and no problems arise as a result of any mismatch between the scan time and the decode time of the reader. The prior art and the instant reader perform equally well under such light decoding loads as will be seen later from Fig. 13.

[000160] Under heavy decoding loads, however, prior art methods do not allow sufficient time for decoding. Thus, as shown in Fig. 12B, when a first scan, Scan 1 is completed, a second scan, Scan 2 is initiated immediately. Scan 2 is then followed by Scan 3 while the decoding of Scan 1 is still in progress. As this situation continues, the decoding process will be seen to fall further and further behind the scanning process until, at some point, the data memory becomes filled. When this occurs new scan data will overwrite old scan data which was not processed, thereby causing a loss of large blocks of scan data.

[000161] In the embodiment disclosed in prior copending Application No. 08/205,539, now issued as U.S. Patent No. 5,463,214, this problem is solved by modifying the reader in a way that allows the scanning process to be suspended and restarted as required to prevent the decoding process from falling so far behind the scanning process that data overflows the memory and is lost. This embodiment is referred to herein as the "Scan on Demand" or SOD tracking mode. This solution to the problem may be understood with reference to Figs. 13A and 13B. Referring to Fig. 13A, there is shown the operation of the subject embodiment under light decoding loads. It will be noted that, under this condition, the relationship between scanning and decoding is the same as that shown in Fig. 12A.

[000162] Fig. 13B shows the relationship which exists between the scanning and decoding processes when the Scan On Demand mode is used under heavy decoding loads. As shown in Fig. 13B, the suspension of the scanning process continues until the results of the prior scan have been decoded. This prevents the decoding process from falling more than a small amount of time behind the scanning process. As a result, there cannot arise a situation, such as that which can arise with the prior art, in which there is a massive loss of scan data. Because this process is described in detail in U.S. Patent No. 5,463,214, it will not be described in detail herein.

[000163] Referring to Fig. 13C there is shown the tracking relationship which exists between the scanning and decoding operations when these operations are controlled in accordance with a tracking mode referred to as the "Skip Scan" or SS tracking mode. With this mode, under heavy decoding loads, decoding proceeds without interruption so long as the scanning function is called for. As shown in Fig. 13C, each decoding operation begins immediately after the preceding decoding operation ends, and proceeds on the basis of the scan data from the then most current complete block of scan data.

[000164] More particularly, Fig. 13C illustrates one possible scenario in which decoding of Scan 1 data is immediately followed by the decoding of Scan 2 data. This occurs because Scan 3 data is incomplete at the time that the second decoding operation begins. The decoding of Scan 2 data, however, is immediately followed by the decoding of Scan 5 data. This occurs because Scan 5 data represents the then most current complete block of scan data. While the results of scans 3 and 4 are therefore unused and skipped over, the data lost by their non-use is provided by more current scan data or, if decoding is unsuccessful, by the results of a later scan. Any occasional decoding failure that results from the skipping of relatively old blocks of scan data is in any case more than offset by the avoidance of the large scale data losses discussed in connection with Fig. 12B.

[000165] Referring to Fig. 13D there is shown the tracking relationship which preferably exists between the scanning and decoding operations when these operations are performed in a reader which includes two and only two scan data memory spaces A and B. With this reader, the preferred tracking mode is the "Decode on Demand" or DOD tracking mode. With this mode decoding does not proceed without interruption. As shown in Fig. 13D, each decoding operation begins at the beginning of a block of scan data. In the event that the end of a decoding operation does not coincide with the beginning of such a block, *i.e.,* occurs while a scanning operation is still in progress, the beginning of the next decoding operation will be delayed until the scanning operation that is then in progress is completed, and then proceeds with reference to the block of scan data which is produced by that scanning operation.

[000166] More particularly, Fig. 13D shows that the decoding of Scan 1 data is completed while Scan 3 is still in progress, overwriting data for Scan 2. Under this condition, decoding is discontinued for a time period T_{s1} that is equal to the time necessary for Scan 3 to be

completed. At the end of time period T_{S1}, decoding resumes with the then most current block of scan data, namely: the scan data produced during Scan 3. Thus, like the mode which is illustrated Fig. 13C, the mode which is illustrated in Fig. 13D begins its decoding operation with the then most current complete block of scan data.

[000167] Referring to Fig. 13E, there is shown the tracking relationship which exists between the scanning and decoding operations when these operations are performed in a reader which includes three scan data memory spaces A, B and C. With this embodiment decoding proceeds without interruption so long as the scanning function is called for. As shown in Fig. 13E, each decoding operation begins immediately after the preceding decoding operation ends, and proceeds on the basis of scan data from the memory which contains the then most current complete block of scan data.

[000168] More particularly, Fig. 13E shows that the decoding of Scan 1 is completed while Scan 3 is still being acquired. Under this condition, with three memory spaces available, decoding is immediately undertaken on the most recent complete Scan (Scan 2) which is contained in memory space B. Upon the completion of the decoding of Scan 2, decoding is commenced on Scan 4 which is contained in memory space A. Thus, the utilization of three memory spaces allows the decoding portion to be occupied one hundred percent of the time.

[000169] The mode illustrated in Fig. 13C is best suited for use with readers having memories and addressing procedures which can accommodate large numbers of relatively short blocks of scan data having sizes that are not known in advance. Applications of this type typically include readers, such as that shown in Fig. 3, which use 1D image sensors.

[000170] The modes illustrated in Figs. 13D and 13E, on the other hand, are best suited for use with readers having memories and addressing procedures which can accommodate small numbers of relatively long blocks of scan data of fixed length. Applications of these types typically include readers, such as that shown in Fig. 2, which use 2D image sensors. With the embodiment illustrated in Fig. 13D, only two scan data memory spaces are used and decoding is discontinuous. With the embodiment illustrated in Fig. 13E three scan data memory spaces are used and decoding is continuous. More than three scan data memory spaces can also be used if additional decoding resources are made available. The one of these different embodiments which is used in a particular application is a design choice which is based on economic considerations.

[000171] The fact that some embodiments use 1D image sensors while others use 2D image sensors should not be taken to mean that embodiments which use 1D image sensors can only read 1D symbols or that embodiments which use 2D image sensors can only read 2D symbols. This is because techniques exist for using either type of image sensor to read both 1D and 2D symbols. It will therefore be understood that the present reader is not restricted to use with any one type of image sensor or to any one type of bar code or other optically encoded symbol.

[000172] Referring to Fig. 14A, there is shown a memory space M1 suitable for use in storing blocks of scan data of the type produced by a reader with a 1D image sensor, together with a pointer or tracking memory M2 suitable for use in storing address or pointer information that makes it possible for the reader to identify the beginning and end point of a block of interest. As shown in Fig. 14A, the block of scan data produced during a first scan of the target is stored in memory M1 beginning at address SS1 (Scan Start for Scan 1) and ending at address SE1 (Scan End for Scan 1). Similarly, the block of scan data resulting from a second scan of the target is stored between addresses SS2 and SE2, and so on. Because scanning takes place continuously, the end of one scan block (e.g. SE1) coincides with the beginning of the next scan block (*e.g.*, SS2). The sizes (in memory space) of these blocks will ordinarily vary from block to block, depending on the number of data transitions in each 1D scan of the target. The boundaries between blocks will, however, be fixed by the occurrence times of the Scan Interrupt signals which are generated by the image sensor or its clock generating circuitry.

[000173] Locations SS and SE of memory M2 are updated in the course of a series of scans so that they always identify or otherwise point to the address of the beginning and ending of the most recently produced complete block of scan data. As a result, when the decoding circuitry is ready to decode the most recently produced complete block of scan data, it need only refer to locations SS and SE to obtain information as to where to begin and end decoding. Before decoding begins, the contents of locations SS and SE are written into locations DS (Decode Start) and DE (Decode End) so that locations SS and SE can continue to be updated while decoding proceeds on the basis of the contents of locations DS and DE. In the preferred embodiment, the decoding circuitry is programmed to mark these beginning addresses as "invalid" (for example, by changing its sign) after it is acquired. Since the decoding processor is programmed to decode only "valid" data, this assures that it can decode a single block of scan data only once.

[000174] Referring to Fig. 14B there are shown a plurality of memory spaces MA, MB ....MN suitable for use in storing blocks of scan data of the type produced by a reader having a 2D image sensor, together with a pointer or tracking memory MP suitable for use in storing address or pointer information for identifying the memory spaces to be used for entering new scan data, decoding, etc. Since the amount of scan data in each block of scan data is known in advance, being the same for each scan, the starting and ending addresses for each memory space (*e.g.*, A_1 and B_1 and A_N and B_N, etc.) will also be the same for each scan. As a result, the memory to be used for storing new scan data, decoding etc. may be specified by specifying just a few bits stored in memory MP. Location CS, for example, may be used as a pointer which identifies the memory where the current scan is being stored, and location NS may be used as a pointer which identifies where the next scanned image is to be stored.

[000175] Similarly, location CD may be used as a pointer which identifies the memory space where the current decode is being undertaken. Finally, location ND may be used as a pointer which identifies where the next available image is for decoding purposes.

[000176] Under ordinary circumstances, three scan data memory spaces will be sufficient to keep the decoding activity of the reader fully occupied and current. This is because the tracking method allows the skipping over of old blocks of scan data as necessary for the decoder to remain occupied and current. If the decoding load becomes extremely heavy, however, it is possible that more old blocks of scan data are skipped over than is advisable. In such instances, it may be desirable to increase the number of memory spaces from 3 to N, where N may be 4 or even more, and to use more than one decoding circuit. If such an increased number of memories and decoders are used, blocks of scan data may be distributed among the memories according to a simple sequential rule and kept track of by increasing the number of bits in the pointers of memory space MP. In addition, the decoding circuits may be assigned to the then most current complete block of scan data as they become free. It will be understood that all such numbers of memory spaces and decoding circuits and the associated tracking procedure are within contemplation.

[000177] Referring to Fig. 15, there is shown a simplified version of Fig. 6A which eliminates those blocks which do not relate directly to the use of the scanning-decoding

parameters of Fig. 7B to produce decoded output data. Of the blocks shown in Fig. 15, blocks 625, 627, and 646 are common to prior art readers and to readers constructed. The remaining blocks of Fig. 15 operate either singly or in various combinations to establish the permitted combinations of the scanning-decoding modes shown in Fig. 7B. These remaining blocks together comprise the preferred embodiment of the means by which the reader is controlled in accordance with the scanning-decoding relationships called for by the parameter table thereof. Other combinations of flow chart blocks, and other combinations of scanning-decoding parameters may also be used. Blocks 642 and 643 may, for example, be configured so that only a preset number of multiple symbols or a preset number of repeats is permitted. Alternatively, all scanning-decoding control blocks may be collectively replaced by a look-up table which directly specifies the next action to be taken. These and other variants will be understood to be within contemplation.

[000178] In view of the foregoing, it will be seen that the scanning and decoding processes may have a selectable one of any of a plurality of different relationships with one another, some of these relationships being tracking relationships and some being non-tracking relationships. The menuing feature allows a user to select that operating mode, whether or not tracking, which gives the best overall data throughput rate in view of the user's then current objectives.

(b) Autodiscrimination/Code Options

[000179] The manner in which the code options called for by the parameter table are implemented in conjunction with the autodiscrimination feature, will now be described with reference to the flow charts of Figs. 16 and 18. Generally speaking, the flow chart of Fig. 16 illustrates the 1D portion of a complete 1D/2D autodiscrimination process, while the flow chart of Fig. 18 illustrates the 2D portion of a complete 1D/2D autodiscrimination process. If both the 1D and 2D code options of the parameter table are enabled (see options B1 and B2 of Fig. 7B), the steps called for by both Figs. 16 and 18 will be executed before the autodiscrimination process is completed. If, however, only one or the other of the 1D and 2D code options of the parameter table is enabled, only the steps called for by Fig. 16 or by Fig. 18 will be executed before the autodiscrimination process is completed. It will therefore be seen that the menuing features and the autodiscrimination features of the present reader interact with one another in a manner that allows a user to tailor the autodiscrimination circuitry as necessary to achieve the highest possible data throughput rate for a particular application.

[000180] In order to gain an understanding as a whole, it should be borne in mind that the above-described relationships between the decoding and menuing processes exist as a subset of an even more complex set of relationships that include the tracking and multiple symbols features. When, for example, a portion of the flow chart of Figs. 16 and 18 calls for an attempted decode, it must be remembered that the attempted decode takes place in the context of the tracking or non-tracking relationships indicated by the parameter table options. In addition, the number of passes that the processor makes through the flow chart of Fig. 16, before continuing on to the flow chart of Fig. 18, depends upon whether or not the multiple symbols feature has been enabled.

[000181] In principle, at least, each one of the possible combinations of the above-described options may be represented in a complete and separate flow chart and described as such. Because adopting the latter approach would obscure rather than clarify, however, the present application will describe these combinations simultaneously in terms of a representative flow chart, with different options being described potential variants of that representative flow chart.

[000182] Turning first to the flow chart of Fig. 16, there is shown the 1D portion of the autodiscrimination process, which operates on a set of image data that has been scanned from a target symbol of unknown type and orientation and stored in RAM 45. If the reader is a 2D reader, this image data will comprise a gray scale representation of the 2D image formed on the image sensor, each pixel of the image sensor being represented by an image data element that includes an 8 bit gray scale indication of its brightness. If, on the other hand, the reader is a 1D reader, the image data may comprise either binary or gray scale values.

[000183] If the reader includes a 2D image sensor, this image data will have been scanned as a 2D image while the reader is held substantially stationary with respect to its target. If the reader includes a 1D image sensor this image data will have been scanned as a series of 1D images while the reader is being moved asynchronously across the target in the manner described in copending commonly assigned U.S. Patent Application No. 08/504,643, now U. S. Patent No. 5,773,806, which is expressly incorporated herein by reference.

[000184] On encountering block 1605, the processor is directed to calculate the "activities" of selected image data elements. The "activity" of a point P as used herein comprises a measure of the rate of change of the image data over a small two dimensional portion of the region surrounding point P. This activity is preferably calculated along any two arbitrarily selected directions which are mutually perpendicular to one another, as shown by the lines parallel to directions X and Y of Fig. 17A. One example of an activity calculation is that which is based on the squares of the gray scale differences of two pairs of points P1X - P2X and P1Y - P2Y that are centered on point P, as shown in Fig. 17A. Two mutually perpendicular directions are used because the orientation of the symbol is unknown and because a high activity level that by chance is difficult to detect in a first direction will be readily detectable in a second direction perpendicular to that first direction.

[000185] In the preferred embodiment, an activity profile of the image data is constructed on the basis of only a selected, relatively small number of image data elements that are distributed across the field of view that corresponds to the stored image data. Using a relatively small number of data elements is desirable to increase the speed at which the symbol may be imaged. These selected points may be selected as the points which lie at the intersections of an X-Y sampling grid such as that shown in Fig. 17A. The spacing of the lines defining this grid is not critical, but does affect the resolution with which the activity profile of the image can be measured.

[000186] When the processor has determined the activities of the selected image data points, it is directed to block 1610, which causes it to look for candidate bar code symbols by identifying regions of high activity. This is conveniently done by determining which sets of image data points have activities that exceed a predetermined activity threshold value. A simplified, one-dimensional representation of this step is illustrated in Fig. 17B, wherein those image data points having an activity that exceed a threshold value TH are labeled as a candidate symbol region CSR1.

[000187] In embodiments which are adapted to find and decode all of the symbols that occur in fields of view that include a plurality of bar code symbols, (*i.e.*, embodiments in which the multiple symbols option is enabled), the result of the step called for by block 1610 is the identification of a plurality of candidate symbol regions (CSRs), any one or more of which may be a bar code symbol. Whether or not they are bar code symbols is determined on the basis of whether they are decodable. As will be explained more fully later, if the multiple symbols option is not enabled, the processor may be instructed to select one of the CSRs according to a suitable selection rule, such as the largest CSR first, the CSR nearest the center of the field of view first, the CSR with the highest total activity first, etc., and then attempt to decode only that symbol and stop, whether or not a symbol has been decoded. Alternatively, as a further option, the processor may be instructed to attempt to decode each CSR in turn until one of them is successfully decoded, and then stop. If the multiple symbols option is enabled, the processor will process all of the CSRs, in turn, according to a suitable priority rule, and continue to do so until all of the CSRs have been either decoded or have been determined to be undecodable.

[000188] Once all CSRs have been located, the processor is directed to block 1615, which calls for it to select the then largest (or most centrally located) as yet unexamined CSR for further processing, and then proceed to block 1620. The latter block then causes the processor to find the centroid or center of gravity of that CSR, before proceeding to block 1625. An example of such a centroid is labeled C in Fig. 17C. Because the steps involved in finding a centroid are well known, they will not be described in detail herein.

[000189] On encountering block 1625, the processor is directed to examine the selected CSR by defining various exploratory scan lines therethrough, determining the activity profile of the CSR along those scan lines, and selecting the scan line having the highest total activity. In the case of a 1D bar code symbol, this will be the direction most nearly perpendicular to the direction of the bars, *i.e.*, the optimum reading direction for a 1D symbol.

[000190] On exiting block 1625, the processor encounters blocks 1630 and 1635. The first of these sets a scan line counter to zero; the second defines an initial, working scan line through the centroid in the previously determined direction of highest activity. The result of this operation is the definition, in the image data space representation of the CSR, of a working scan line such as SC=0 in Fig. 17C.

[000191] Once the initial scan line has been defined, the processor is directed by block 1640 to calculate, by interpolation from the image data of the CSR, the values of sampling points that lie along this scan line. This means that, for each sampling point on the initial scan line, the processor will calculate what brightness the sampling point would have if its brightness were calculated on the basis of the weighted brightness contributions of the four nearest measured image data points of the CSR. These contributions are illustrated by the dotted lines which join the sample point SP of Fig. 17D to the four nearest image data points DPA-DPD. So long as these sampling points are more closely spaced than the image data points, this interpolation procedure will be performed on a subpixel basis, and will produce a usably accurate representation of the image data along the scan line. The result of the subpixel interpolation of the sampling points on a representative scan line of this type is shown in Fig. 17E. Because the particulars of the subpixel interpolation process are known to those skilled in the art, this process will not be further described herein.

[000192] Once the above-described scan line data have been calculated, the processor is directed to block 1645, which calls for it to binarize the scan line data, *i.e.,* convert it to a two-state representation of the data which can be processed as a candidate for 1D decoding. One such representation is commonly known as a timercount representation. One particularly advantageous procedure for accomplishing this binarization process is disclosed in U.S. Patent No. 5,286,960, which is hereby incorporated herein by reference.

[000193] On exiting block 1645, the processor will be in possession of a potentially decodable two-state 1D representation of the CSR. It then attempts to decode this representation, as called for by block 1650. This attempted decoding will comprise the trial application to the representation of one 1D decoding program after another until the latter is either decoded or determined to be undecodable. Because decoding procedures of the latter type are known to those skilled in the art, they will not be discussed in detail herein.

[000194] As the 1D autodiscrimination process is completed, the processor is directed to decision block 1655 which causes it to continue along one of two different paths, depending on whether or not decoding was successful. If it was not successful, the processor will be caused to loop back to block 1635, via blocks 1660 and 1665, where it will be caused to generate a new working scan line that is parallel to initial scan line SC=0, but that passes above or below centroid C. This looping back step may be repeated many times, depending on the "spacing" of the new scan lines, until the entire CSR has been examined for decodable 1D data. If the entire CSR has been scanned and there has been no successful decode, the processor is directed to exit the just-described loop via block 1670. As used herein, the term "parallel" is used in its broad sense to refer to scan lines or paths which are similarly distorted (*e.g.,* curvilinear) as a result of foreshortening effects or as a result of being imaged from a non-planar surface. Since compensating for such distorting effects is known, as indicated, for example, by U.S. Patent No. 5,396,054, it will not be discussed in detail herein.

[000195] Block 1670 serves to direct the processor back to block 1615 to repeat the above-described selection, scanning and binarizing steps for the next unexamined CSR, if one is present. If another CSR is not present, or if the processor's program calls for an attempt to decode only one CSR, block 1670 causes the processor to exit the flow chart of Fig. 16 to begin an attempt to decode the then current set of image data as a 2D symbol, in accordance with the flow chart of Fig. 18. If other CSRs are present, and the multiple symbols option is enabled, block 1670 directs the processor back to block 1615 to repeat the selection, scanning and binarizing process for the next CSR, and the next, and so on, until there is either a successful decode (block 1655) or all of the CSRs have been examined (block 1670).

[000196] If the processing of the first CSR has resulted in a successful decode, block 1655 directs the processor to block 1675, which causes it to determine whether the decoded data indicates that the CSR contains a 1D stacked symbol, such as a PDF417 symbol. One example of such a symbol is shown in Fig. 19D. If it is not, *i.e.,* if the decoded symbol includes only a single row of bars, the 1D data is stored for later outputting in accordance with block 648 of the main program of Fig. 6A, as called for by block 1680. Alternatively, the data may be output immediately and block 648 later skipped over. Then, if there are no remaining unexamined CSRs, or if the multiple symbols option is not enabled, the processor is directed to exit the flow chart of Fig. 16 via block 1682. If, however, there are remaining CSRs and the multiple symbols option is enabled, block 1682 will direct the processor back to block 1615 to begin processing the next CSR, and the next, and so on until all CSRs have been examined and decoded (block 1682) or examined and found to be undecodable (block 1670).

[000197] If, on encountering block 1675, the decoded data indicates that the CSR contains a 1D stacked symbol, the above-described processing is modified by providing for the repetition of the scanning-digitizing process, beginning with block 1635. This is accomplished by blocks 1684, 1686 and 1688 in a manner that will be apparent to those skilled in the art. Significantly, by beginning the repeating of the process at block 1635, all additional scan lines defined via the latter path will be parallel to the first decodable scan line, as required by a 1D stacked symbol, at least in the broad sense discussed earlier.

[000198] In view of the foregoing, it will be seen that, depending on the number of CSRs that have been found in the stored image data, and on the enablement of the multiple symbols option, the flow chart of the embodiment shown in Fig. 16 will cause all 1D symbols in the image data to be either decoded or found to be undecodable before directing the processor to exit the same.

[000199] As will be explained more fully in connection with Fig. 20, the 2D autodiscrimination flow chart of Fig. 18 may be processed after the processing of the 1D autodiscrimination flow chart of Fig. 16 has been completed. It may also be processed without the flow chart of Fig. 16 having been previously processed, *i.e.,* the 1D portion of the 1D/2D autodiscrimination process may be skipped or bypassed. (In principle, the steps of the 2D portion of the 1D/2D autodiscrimination process (Fig. 18) may also be processed before the 1D portion thereof (Fig. 16), although this option does not comprise the preferred embodiment). This is because the code options of the menuing feature make all of these options selectable by the user. It will therefore be understood that the present feature contemplates all possible combinations of autodiscrimination options.

[000200] Referring to Fig. 18, there is shown a flow chart of the 2D portion of the 1D/2D autodiscrimination process. When the flow chart of Fig. 18 is entered, the image data that is stored in RAM 45 is the same as that which would be stored therein if the flow chart of Fig. 16 were being entered. If the reader is a 2D reader this image data will comprise an array of 8-bit gray scale image data elements produced by image sensor 32-2 and its associated signal processing and A/D converter circuits 3502 and 36-2. If the reader is a 1D reader that produces a 2D image by being moved across the target symbol, the image data will comprise an array of binary data elements such as those shown in above-cited copending Application No. 08/504,643, now U. S. Patent No. 5,773,806.

[000201] The flow chart of Fig. 18 begins with a block 1805, which directs the processor to convert the gray scale image data representation stored in RAM 45 (if present) into a two-state or binarized representation of the same data. This may be accomplished in generally the same manner described earlier in connection with Fig. 17B, *i.e.,* by comparing these gray scale values to a threshold value and categorizing these values as 1s or 0s, depending upon whether they exceed or do not exceed that threshold value.

[000202] Once the image data has been binarized, the processor continues on to block 1810, which causes it to identify and locate all of the 2D finder patterns that appear in the field of view of the image data. This is preferably accomplished by examining all of the candidate 2D finder patterns (CFPs) that are present and identifying them by type, *i.e.,* identifying whether they are bullseye type finder patterns, waistband type finder patterns or peripheral type finder patterns. An example of a bullseye type finder pattern is shown in the central portion of the 2D bar code symbol of Fig. 19A, which symbol encodes data in accordance with a 2D matrix symbology named "Aztec." An example of a waistband type finder pattern is shown in the middle portion of the 2D bar code symbol of Fig. 19B, which symbol encodes data in accordance with a 2D matrix symbology named "Code One". An example of a peripheral type finder pattern is shown in the left and lower edges of the 2D bar code symbol of Fig. 19C, which symbol encodes data in accordance with a 2D matrix symbology known as "Data Matrix." The finder identification process is performed by applying to each CFP, in turn, a series of finder pattern finding algorithms of the type associated with each of the major types of finder patterns. Since such finder finding algorithms are known for finders of the waistband and peripheral types, these algorithms will not be discussed in detail herein. One example of a finder finding algorithm for a waistband type finder, may be found, for example, in "Uniform Symbology Specification Code One", published by AIM/USA Technology Group. Finder finding algorithms for bullseye type finders that include concentric rings, (e.g. MaxiCode) are also known and will also not be described in detail herein.

[000203] Particularly advantageous for purposes, however, is bullseye type finder finding algorithm of the type that may be used both with 2D symbologies, such as MaxiCode, that have bullseye finder patterns that include concentric rings and with 2D symbologies, such as Aztec, that have bullseye finder patterns that include concentric polygons. A finder finding algorithm of the latter type is described in copending, commonly assigned U.S. Patent Application No. 08/504,643, now U. S. Patent No. 5,773,806, which has been incorporated herein by reference. The Aztec 2D bar code symbology itself is fully described in U.S. Patent Application No. 08/441,446, which has also been incorporated herein by reference.

[000204] Once all of the finder patterns have been located and their types have been determined, the processor is directed to decision block 1815. This block affords the processor an opportunity to exit the flow chart of Fig. 18, via exit block 1820, if no 2D finder patterns could be found and typed. This block speeds up the execution of the program by skipping over decoding operations which have no hope of success without their associated finder pattern.

[000205] If a finder pattern has been found and typed, the processor is directed to block 1825. This block causes the processor to select for decoding the bar code symbol whose finder is closest to the center of the field of view of the image data. Optionally, the processor may be instructed to find the largest 2D bar code symbol that uses a particular 2D symbology or the 2D bar code symbol using a particular 2D symbology which is closest to the center of the field of view of the image data. The "closest-to-the-center" option is ordinarily preferred since a centrally located symbol is likely to be a symbol, such as a menu symbol, at which the user is deliberately aiming the reader. Once this selection has been made, the processor attempts to decode that symbol, as called for by block 1830. If this decoding attempt is successful, as determined by decision block 1835, the resulting data may be stored for outputting in accordance with block 648 of the main program of Fig. 6A, as called for by block 1840. Alternatively, the decoded data may be output immediately and block 648 later skipped over. If the decoding attempt is not successful, however, block 1840 is skipped, and the processor is directed to decision block 1845.

[000206] If the user has elected not to use the multiple symbols option, block 1845 may direct the processor to exit the flow chart of Fig. 18, via block 1850, after any 2D symbol has been successfully decoded. Optionally, block 1845 may be arranged to direct the processor to exit the flow chart of Fig. 18 after the attempted decoding of the centermost symbol, without regard to whether or not the decoding attempt was successful.

[000207] If the user has elected to use the multiple symbols option, block 1845 will direct the processor back to block 1825 to process the next 2D symbol, *i.e.,* the symbol whose CFR is next closest to the center of the field of view. The above-described attempted decoding and storing (or outputting) steps will then be repeated, one CFR after another, until there are no more symbols which have usable finder patterns. Finally, when all symbols having usable finder patterns have been either decoded or found to be undecodable, the processor will exit the flow chart of Fig. 18, via block 1850, to return to the main program of Fig. 6A.

[000208] In view of the foregoing, it will be seen that, depending on the number of identifiable CFRs that have been found in the stored, digitized image, and on the enablement of the multiple symbols option, the 2D autodiscrimination routine shown in Fig. 18, will cause one or more 2D symbols in the image data to be either decoded or found to be undecodable before directing the processor to exit the same.

[000209] For the sake of clarity, the foregoing descriptions of the 1D and 2D phases of the 1D/2D autodiscrimination process have been described separately, without discussing the combined or overall effect of the code options and scanning-decoding options discussed earlier in connection with Fig. 7B. The overall effect of these code options and the manner in which they are implemented will now be described in connection with Fig. 20. As will be explained presently, Fig. 20 shows (with minor simplifications) the contents of block 627 of Fig. 6A. It also shows, as blocks 2010 and 2035 (again with minor simplifications), the 1D and 2D autodiscrimination routines discussed earlier in connection with Figs. 16 and 18, respectively.

[000210] On entering the flow chart of Fig. 20, the processor encounters a block 2005 which causes it to determine, with reference to the code options of the parameter table, whether all of the 1D codes have been disabled. If they have not, the processor continues to block 2010. In accordance with block 2010, the processor performs the 1D autodiscrimination process described in connection with Fig. 16, using the1D code and scanning-decoding options indicated by the parameter table. Depending upon whether 1D decoding was successful, as determined by block 2015, the processor either outputs (or stores) data per block 2020 and exits, or continues on to blocks 2030 and 2035 to begin the 2D autodiscrimination process.

[000211] If all 1D codes have been disabled, the processor is directed directly to block 230, thereby skipping block 2010 in its entirety. Then, unless all 2D codes have also been disabled (per block 2030), it proceeds to block 2035 to begin the autodiscrimination process described in connection with Fig. 18, using the 2D codes and scanning-decoding options indicated by the parameter table. Depending upon whether 2D decoding was successful, as determined by block 2040, the processor either outputs (or stores) data, per block 2045, or returns to the main program of Fig. 6A. Returning to the latter then causes or does not cause further scans to be made depending on the states of blocks 635 and 640 thereof.

[000212] In view of the foregoing, it will be seen that the 1D/2D autodiscrimination process may be practiced in many different ways, depending upon the menuing options that have been chosen by the user. Among these menuing options, the code options increase the data

throughput rate of the reader by assuring that the processor does not waste time trying to autodiscriminate and decode symbols which it has been told are not present, or are not of interest. The scan tracking options also increase the data throughput rate of the reader by assuring that the scanning and decoding phases of read operations both operate, to the extent possible in view of the then current decoding load and decoding options, at a 100% utilization rate. Even the multiple symbols option also increases the data throughput rate of the reader by either discontinuing the reading of symbols that are not centered and therefore not of interest or speeding up the processing of multiple symbols that are of interest. Thus, for a processor with a given performance rating and a set of decoding programs of given length, the apparatus assures a higher overall data throughput rate than has heretofore been possible.

[000213] There is provided an optical scanning and decoding apparatus and method, suitable for use with bar code readers, bar code scanning engines, and portable data terminals (PDTs), which combines improved scanning-decoding and autodiscrimination features in the context of an apparatus and method which also provides improved menuing and reprogramming features.

[000214] In accordance with the menuing feature, there is provided an improved apparatus and method which enables a user to determine the current operating mode of an optical reading apparatus, and to rapidly and conveniently change that operating mode to optimize it for operation under then current conditions. The menuing feature, for example, enables the user, via a machine readable table of pre-recorded menu symbols, to command the reader to communicate with a host processor using one of a number of protocols, to command the reader to format the decoded output according to host processor requirements, or to command the reader to report to the host processor any of a plurality of types of information about the current operating state of the reader, such as the version of software then being used, the code options that are then being used, and even a complete listing of the reader's parameter table. If a suitable printer is available, the complete status of a first reader may be output as a machine readable menu symbol that other, similarly equipped readers may read and use to reconfigure themselves for operation in the same manner as the first reader.

[000215] In accordance with the reprogramming feature, there is provided an improved apparatus and method by which an optical reader may be reprogrammed from a source external to the reading apparatus, with or without the participation of a user. This external source may be either on-site, *i.e.,* located at the same local facility as the reader, or off-site, *i.e.,* located at a remote facility that is coupled to the local facility only via a transmission line or computer network. When actuated, the reprogramming feature enables a reader to reprogram itself, either in whole or in part, and thereby become able to operate with operating software of the latest type. Depending on the application, the reprogramming of the reader may be initiated either by a host processor external to the reader, as by a command issued via the reader's communication port, or by a user initiated command issued as a part of the above-mentioned menuing process.

[000216] In accordance with another aspect of the reprogramming feature, a local host processor may be configured to carry out reprogramming of an optical reader or another type of portable data terminal. In a reprogramming subroutine a local host processor can be made, at the selection of a user, to replace an entire main program and parameter table of a reader, or else one of either a main program or a parameter table of an operating program individually.

[000217] In accordance with another subprogram of a local host processor, the local host processor can be made to edit a parameter table. When this subprogram is selected the user may either edit the parameter table that is stored in a memory device of the reader or else edit a parameter table stored in a memory device in communication with the local host processor. After editing, the user may write the edited parameter table to the reader's memory device, write the edited parameter to a bulk storage device for later use, or print or display the edited parameter table.

[000218] In accordance with another aspect, an optical reader may be made to receive a component control instruction from a host processor which is transmitted in response to a user input command to remotely control an optical reader. In accordance with this aspect, the optical reader is made to execute a component control instruction substantially on-receipt thereof. In one embodiment, execution by an optical reader of a component control instruction has the same effect as a reader trigger being manually pulled.

[000219] There is also provided an optical scanning and decoding apparatus and method which includes improved scanning-decoding and autodiscrimination features, either or both of which may be used in conjunction with, and/or under the control of, the above-described menuing and reprogramming features. In other words, the autodiscrimination feature is made available to the user on a menu selectable or reprogrammable basis to speed up and/or update the decoding phase of the scanning and decoding process. Together, these features enable the reading apparatus to read and decode a wide range of optically encoded data symbols at an improved data throughput rate.

[000220] When a reader is one in which the scan engine cannot be readily started and stopped, or in which such starts and stops impose unacceptable delays or produce user perceptible flicker, preferably operates in one of the tracking relationships described in previously mentioned copending Application No. 08/914,883, now U. S. Patent No. 5,942,741. One of these tracking relationships is a Skip Scan tracking relationship in which the results of one or more scans may be skipped over entirely in favor of more recently produced scan results. Another is a Decode On Demand tracking relationship in which decoding is suspended briefly as necessary to allow a scan then in progress to be completed. The latter relationship is ordinarily not preferred, but is still useful when the reader is such that its scan memory is able to store only two complete blocks of scan data.

[000221] When the reader is one in which the scan engine can readily be stopped, the present reader may operate in the tracking relationship described in previously mentioned U.S. Patent No. 5,463,214. With this, "Scan On Demand" tracking relationship, scanning is suspended briefly as necessary to prevent scanning and decoding from becoming uncorrelated with one another.

[000222] In the preferred embodiment, the reader includes an algorithm that is able to accommodate any of the above-described scanning-decoding relationships, among others. Which of them is actually used will vary from reader to reader depending upon the size and type of memory and the type of scan engine used thereby, and may be changed from time to time.

[000223] The present reader also contemplates and provides for at least one scanning-decoding relationship which does not fall within the meaning of the above-defined tracking relationships. One of these non-tracking relationships is a "One Shot" relationship or mode in which a single scan is followed by a single decoding attempt and then a stoppage. Such scanning-decoding events may be initiated by respective single actuations of a manual trigger. Because of its inherently discontinuous nature, the use of the One Shot mode implies the non-use of any of the above-mentioned tracking modes.

[000224] Two other such scanning-decoding relationships are referred to herein as the "Repeat Until Done" relationship or mode and the "Repeat Until Stopped" relationship or mode. With the Repeat Until Done relationship, scanning and decoding operations follow one after another until a successful decode occurs, and are then discontinued. With the Repeat Until Stopped relationship, scanning and decoding operations follow one after another and continue, even after sets of decoded data are stored or output, until instructed to stop by the release of the trigger or by the readers' program. Because of their repetitive nature, the use of Repeat Until Done and Repeat Until Stopped modes are usable both in conjunction with the above-described tracking modes and independently of those tracking modes. As a result, the Repeat Until Done and Repeat Until Stopped modes may be implemented as user selectable non-tracking relationships or as tracking relationships.

[000225] In embodiments that use the auto discrimination feature, there is provided a method and apparatus by which a plurality of different symbols of a multiplicity of different types may be scanned and decoded in a manner that is optimized for a particular application, on either a menu selectable or a reprogrammable basis. When all of the symbols to be autodiscriminated are known to be 1D symbols, for example, the data throughput rate may be increased by structuring the autodiscrimination feature so that no attempt is made to decode 2D symbols, or vice versa. When, on the other hand, the symbols to be autodiscriminated are known to all be of (or all not to be of) a few types, whether 1D or 2D, the data throughput rate may be increased by structuring the autodiscrimination feature so that all but a few (or only a few) 1D and/or 2D symbologies are disabled, *i.e.,* so that no attempt is made to decode them. Other possible autodiscrimination options include not decoding or not outputting data for symbols that encode messages that are too long or too short to be of interest in a particular application. Any of these options may be chosen and changed as necessary to achieve the highest possible data throughput rate.

[000226] Because of the large number of different combinations of distinct operational states that are made possible thereby, the apparatus and method will be seen to have a protean quality that not only makes it usable in a large number of different applications, but also enables it to continue to remain so usable as new functions, new bar code symbologies and new and updated decoding programs are developed in the future.

[000227] While the present invention has necessarily been described with reference to a number of specific embodiments, it will be understood that the time spirit and scope should be determined only with reference to the following claims.